~~A~~ Using the COAsT Python Package to Develop a Standardised Validation ~~Framework~~ Workflow for Ocean Physics Models~~:~~ ~~Application to the Northwest European Shelf~~

David Byrne[1], Jeff Polton[1], Enda O' Dea[2], and Joanne Williams[1]

[1]National Oceanography Centre, Liverpool, UK
[2]Met Office, Exeter, UK

Correspondence: David Byrne (dbyrne@noc.ac.uk)

Abstract. Validation is one of the most important stages of a model's development. By comparing outputs to observations, we can estimate how well the model is able to simulate reality, which is the ultimate aim of many models. During development, validation may be iterated upon to improve the model simulation and compare to similar existing models or perhaps previous versions of the same configuration. As models become more complex, data storage requirements increase and analyses improve, scientific communities must be able to develop standardised validation workflows for efficient and accurate analyses with an ultimate goal of a complete, automated validation.

We ~~set out our process and principles used to construct~~ describe how the COAsT Python package has been used to develop a standardised and partially automated validation system. This is discussed alongside five principles which are fundamental for our system: system scaleability, independence from data source, reproducible workflows, expandable code base and objective scoring. We also describe the current version of our own validation workflow and discuss how it adheres to the above principles. ~~We use the COAsT Python package as a framework within which to build our analyses. COAsT~~ COAsT provides a set of standardised oceanographic data objects ideal for representing both modelled and observed data. We use the package to compare two model configurations of the Northwest European Shelf to observations from tide gauge and profiles.

1 Introduction

~~Modelling~~ Numerical modelling plays a vital part in both the prediction and understanding of ocean processes. Models must be validated in order to determine their accuracy. Generally, this is done by comparing their output to analogous observed data or data sets derived from observations with the aim of quantifying how close model simulations are to the reality they attempt to replicate. The reason for this is clear: operational forecasting models must be accurate for the communities receiving predictions, and scientific simulations must be able to adequately represent the processes we seek to study.

New and existing model configurations must go through a development process during which input data sets are defined and refined, physical parameterizations are chosen and their parameter values are tuned. The repeated validation required during this iterative process is one of the most important stages of model development. However, it can often be time and resource consuming, lack consistency and performed in a new and arbitrary fashion for subsequent model configurations.

In this paper, we introduce ~~a flexible and standardised offline model validation workflow which we have developed for our own development processes and we discuss some of the most important guiding principles when developing such a system.~~ and demonstrate the validation and observation classes within the Coastal Ocean Assessment Toolbox (COAsT) Python package. This package can be used to standardize and simplify workflows for validation of ocean model data. We describe the relevant components of COAsT alongside our philosophy and key principles behind our validation workflows. Some of these ideas are demonstrated as we set out our workflow for validation of temperature and salinity again ocean profiles and sea surface height against tide gauge data.

~~Prior to development of a validation workflow it is important to establish foundational principles with which to work. For our workflow, these are as follows:~~

1. ~~**Scales with size of data**~~

   ~~As the power of computing resources increases, model configurations are being developed with higher resolutions and smaller time steps, amplifying the technical challenges associated with analysis and validation. A long-term problem is the storage of data, which must be tackled through careful choices of output and file compression. However, a more immediate issue is how to analyse this data, when there is no hope of being able to fit it all in a processor's memory. For a validation system to be truly flexible, it should be able to scale easily alongside these increasing data sizes. Fortunately, computer science offers many potential solutions to this problem, including ideas such as lazy loading, data chunking and parallel programming. All three of these concepts involve systematically and iteratively dealing with smaller portions of the whole dataset, potentially over multiple processors, in order to analyse large datasets.~~

2. ~~**Data source independence**~~

   ~~Validation code should work quickly and easily with modelled or observed datafrom various input sources. For example, performing an analysis of output from different numerical models or making a comparison between model data and observation datasets from different sources and in different input formats.~~

3. ~~**Reproducible and citable**~~

Ensuring that validation analyses are easily reproducible means that the same workflow can be applied quickly (or automatically) to an ensemble of model runs or different configurations. This can save time and resources during the phase of model development where physical parameterizations are being chosen and tuned. In addition, this aids in knowledge and code sharing with other researchers who wish to perform an equivalent analysis on their own data. Having reproducible code is not only important for the current state of a code base, but also for all of its history. In other words, it may be important to rerun a historical analysis using the validationworkflow, however the code may have since been modified. We can ensure that we use the equivalent code by using a version control system alongside the validation codebase and publishing periodic release versions. Each release version may be citable using, for example, a DOI, making it much easier to use historical code where necessary.

4. **Expandable codebase**

No analysis is static and will change over time, with modifications to existing analyses and the addition of new metrics and methods. Indeed the validation presented in this paper is only a foundation and has many avenues for expansion. It is therefore vital that code is written in such as way as to make changes and expansions by a community of contributors as easy as possible. For example, using programming structures such as classes, making the code public and open source and ensuring that there are good programming practices and conventions in place. This can be further aided by having well written documentation. This point also reiterates the need for repeatability via version control and citations.

5. **Objective scoring**

Deriving metrics which minimize room for interpretation is essential when comparing large numbers of model configurations. Where possible, a validation should not be done subjectively, for example visual comparisons of time series. Instead, well defined scoring metrics should be used. This might include commonly used integrated metrics such as the Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and correlation or more complicated probabilistic metrics such as the Continuous Ranked Probability Score (CRPS) (Matheson and Winkler, 1976). Some types of analyses also require different approaches such as extreme value analyses (e.g event based hazard validation).

We can satisfy many of the above principles using COAsT (Coastal Ocean Assessment Toolbox): a Python package with highly useful standardised oceanographic data structures and routines. COAsT also acts as a framework for building repeatable and shareable oceanographic analysis code. All of the analysis presented in this paper is available within the COAsT package and we discuss this in In the following subsections, we provide an overview of COAsT, and its relevance to five key principles applicable to ocean model validation. Then, in Sections 3 – 4, we showcase the *Gridded*, *Profile* and *Tidegauge* data classes, as shown in Table-2. The *Gridded* class is used to read, represent and manipulate output from two NEMO model runs (see Section-1.1.

In the following sections we describe some of the decisions made in the initialisation of our validation workflow. In **??**). We then use this object interactively with the *Profile* and *Tidegauge* classes to do a comparison between the model and observed data. See Section-2 we provide 2.2 for more information on the COAsT package alongside details on the model and observed

~~data usedin this paper. In Sections 3 – 4, we showcase some of the analyses available within the package and present results for tidal data, non-tidal residuals and tracers (temperature and salinity)respectively. Finally, in Section-5, we discuss the future of this workflow and opportunities for its expansion~~ observation data used.

2 ~~Methods~~

The package is open source and all of the code is freely available via Github (www.github.com). This again satisfies the validation principles set out in the previous section. Using COAsT provides a level of transparency which aids knowledge sharing and discussion as well as user contribution and expansion of the workflow.

1.1 The COAsT Framework

COAsT ~~(Coastal Ocean Assessment Toolbox)~~ is a Python package and framework that aims to standardise ~~the analysis and comparison~~ many of the aspects of the analysis of modelled and observed oceanographic datasets. ~~In building a validation workflow, COAsT provides us with the perfect environment in which to satisfy the criteria set out in the previous section.~~ At its core, the package provides the user with a set of standardised data ~~objects~~ classes which are designed for oceanographic analyses. By using these standardised structures, a user can be sure that analysis code will work for their data, regardless of source, so long as it adheres to the appropriate structure. The package builds upon key libraries including Xarray (Hoyer and Hamman, 2017) and Dask (Rocklin, 2015). The dataset contained within each instance of a COAsT data class is an Xarray dataset, which is a structured and labelled multidimensional array. Depending on the class in question, the names, structure and layout of dimensions and variables are enforced and controlled by a .json file, which can optionally be modified.

~~To demonstrate,~~ As an example, the Gridded class is designed to be used with model data on a regular grid and must contain an xarray Dataset that has dimensions *t_dim* (time), *z_dim* (vertical dimension), *y_dim* and *x_dim* (horizontal dimensions). The source of this data could be from a variety of numerical models (for example NEMO, ROMS or similar) and data could be stored in any xarray compatible file (e.g. netCDF, zarr). Once data has been ingested into the COAsT framework, the same validation script can be more easily applied repeatedly as dimension names and variable names will be known and any necessary preprocessing steps will have been taken.

Now that we have loaded our model data into the framework, suppose we wish to apply an analysis to temperature data taken from vertical profiles of the ocean, e.g. observations from CTD casts. As discussed later in the paper, COAsT provides a *Profile* data class for this purpose. Each instance of a *Profile* object can contain a dataset representing multiple profiles. This dataset must adhere to a predefined structure and naming conventions, specifically an index dimension called *id_dim* and a vertical dimension called *z_dim*. ~~Then the analysis may be written with this structure in mind, not the native structure of the data source or file format. By providing this middle layer into the workflow, it is much simpler to apply the analysis to multiple data sources, to share it with others and to expand upon it in the future.~~

~~COAsT builds upon a few key libraries, including Xarray (Hoyer and Hamman, 2017) and Dask (Rocklin, 2015). The dataset contained within each instance of a COAsT dataclass is an Xarray dataset. This has many of the attributes of a netCDF file,~~

but within the Python environment. This includes multidimensional arrays with named dimensions, coordinates and multiple variables. Xarray is also very useful for reading and manipulating files in the netCDF format and by using it in combination with Dask, the user has access to a powerful system that provides lazy loading, chunking and parallel code. As discussed in the previous section, these are important ideas when handling very large datasets. Lazy loading (or asynchronous loading) means that data is not brought to memory until needed. It works well with chunking, where an analysis is performed on smaller blocks of data rather than the whole dataset at once. Chunks of the data can be analysed either in series or in parallel by utilising multiple CPUs. If a user has profile data from different sources, then they can be easily combined within the COAsT system. The model data, represented using the Gridded class, can then be compared to the data contained within the Profile class.

The COAsT package uses an object-oriented approach, and classes can be broadly separated into two types: data and analysis classes. Data classes are those discussed above, wherein the structured datasets are read, stored and manipulated. Data classes are an ideal place to keep manipulation or visualization routines, e.g. subsetting data based on some criteria or plotting some known variable, as the class is aware of the structure of the dataset it contains. Analysis classes are much more flexible more flexible and general, with no real constraints. They contain code for analysis of one or multiple data objects. This object oriented approach is ideal for expansion of the package itself, facilitating contributions from the scientific communityAlternatively, analyses can be contained within external scripts or libraries, which make use of the fundamental COAsT structures.

At the time of writing, the data classes have two main parent classes, from which all others are derived: the *Gridded* class and *Indexed* class. These are summarised in Table-1. The *Gridded* class enforces a data structure ideal for output from structured models such as NEMO[1]. It has horizontal dimensions *x_dim* and *y_dim*, a vertical dimension *z_dim* and time dimension *t_dim*. It labels these dimensions with 1-dimension 1-dimensional time coordinates, 2-dimensional latitude and longitude coordinates and 3-dimensional depth coordinates. The *Indexed* class is a general class for data that can lie along a single index, such as the *Profile* data discussed earlier. It has a generalised dimension structure with an index dimension *id_dim*, a time dimension *t_dim* and a vertical dimension *z_dim*. The children of this class have different permutations of these dimensions, and they are summarised in Table-2.

The package is open source and all of the code is freely available via Github ()

1.2 Five Principles for Validation

There are five key principles that any validation workflow should adhere to and integrating COAsT into a workflow can help satisfy them:

1. **Scales with size of data**

   As the power of computing resources increases, model configurations are being developed with higher resolutions and smaller time steps, amplifying the technical challenges associated with analysis and validation. A long-term problem is the storage of data, which must be tackled through careful choices of output and file compression. However, a more immediate issue is how to analyse this data, when there is no hope of being able to fit it all in a processor's memory. For

[1]https://www.nemo-ocean.eu (last accessed 28 June 2022)

Table 1. The two main data classes in the COAsT package and their function. The *Gridded* class is used in this paper to represent output data from the NEMO model. Child classes of the Indexed class are used to represent observed data (see Table-2)

Class	Data Structure	Function
Gridded	– **Dimensions:** (t_dim, z_dim, ~~x~~y_dim, ~~y~~x_dim) – **Coordinates:** time, depth, longitude, latitude	For representing gridded data such as from a structured model or reanalysis dataset. Contains additional routines for reading some file formats and manipulating data in space and time.
Indexed	– **Dimensions:** (id_dim, z_dim, t_dim) – **Coordinates:** time, longitude, latitude, depth	For representing data that can lie along a single index. Can be used for many observation types and has many subclasses for different instruments such as *Profile*, *Tidegauge* and *Glider*. These additionally contain routines for reading from some common observation databases as well as manipulation of datasets in time and space.

a validation system to be truly flexible, it should be able to scale easily alongside these increasing data sizes. COAsT is able to deal with this problem by building upon Python packages such as Xarray (Hoyer and Hamman, 2017) and Dask (Rocklin, 2015), giving the user access to lazy loading[2], data chunking[3] and parallel programming. All three of these concepts involve systematically and iteratively dealing with smaller portions of the whole dataset, potentially over multiple processors, in order to analyse large datasets. ~~This again satisfies the validation principles set out in the previous section.Using COAsT provides a level of transparency which aids knowledge sharing and discussion as well as user contribution and expansion of~~

By pairing it with Dask, the ~~workflow.~~ user has access to a powerful system that provides lazy loading, chunking and parallel code. As COAsT makes use of xarray and Dask, the user also has access to important data analysis concepts such as lazy loading (or asynchronous loading), chunking and parallel computation. Lazy loading means that data is not brought to memory until needed, which works well with chunking, where an analysis is performed on smaller blocks of data rather than the whole dataset at once. Chunks of the data can be analysed either in series or in parallel by utilising multiple CPUs.

~~In Sections 3 – 4, we showcase the *Gridded*, *Profile* and *Tidegauge* data classes, as shown in Table-2. The *Gridded* class is used to read, represent and manipulate output from two NEMO model runs (see~~

2. Data source independence

[2]Lazy loading: https://en.wikipedia.org/wiki/Lazy_loading [last accessed 5 Apr 2023]
[3]An overview on Chunks: https://docs.dask.org/en/stable/array-chunks.html [last accessed 5 Apr 2023]

Table 2. ~~Child classes~~ Four examples of ~~the~~ COAsT class which are children of the Indexed class~~discussed in this paper~~. These are the classes used to represent observed data. Other classes are available within the COAsT package.

Class	Data Structure	Function
Profile	– **Dimensions:** (id_dim, z_dim) – **Coordinates:** id, latitude, longitude, depth, time	For representing one or more vertical oceanographic profiles. Each index is a different set of vertical measurement along a single vertical profile and has a time and location of collection.
Tidegauge	– **Dimensions:** (id_dim, t_dim) – **Coordinates:** time, longitude, latitude	Contains time series data collected at a tide gauge location, e.g. sea level data. This is a child of the *Timeseries* class, which is in turn a child of the *Indexed* class.
Track	– **Dimensions:** (t_dim) – **Coordinates:** time, longitude, latitude	Contains along-track data such as from an altimeter. Data is ordered along a single track, which lies along the *t_dim* dimension.
Glider	– **Dimensions:** (id_dim, t_dim) – **Coordinates:** **time, depth, longitude, latitude**	Contains data along a glider path. Similar to Track, however there is an additional depth coordinate and index dimension.

Validation code should work quickly and easily with modelled or observed data from various input sources. For example, performing an analysis of output from different numerical models or making a comparison between model data and observation datasets from different sources and in different input formats. COAsT allows the user to abstract out the data source by standardizing array structures, dimension names, variable names and basic manipulation routines. Once data has been ingested into COAsT, the source of the data no longer matters and analysis scripts can be applied quickly to any data with compatible structure.

3. **Reproducible and citable**

Ensuring that validation analyses are easily reproducible means that the same workflow can be applied quickly (or automatically) to an ensemble of model runs or different configurations. This can save time and resources during the phase of model development where physical parameterizations are being chosen and tuned. In addition, this aids in knowledge and code sharing with other researchers who wish to perform an equivalent analysis on their own data. Having reproducible code is not only important for the current state of a code base, but also for all of its history. In other words, it may be important to rerun a historical analysis using the validation workflow, however the code may have since

been modified. COAsT uses git for version control alongside a DOI for each release version of the package. This means the exact version of COAsT used for a validation script can be tracked and re used if necessary. Broader application of the reproducibility principle as applied to regional ocean modelling is discussed in more detail in Polton et al. (2023).

4. **Expandable codebase**

   No analysis is static and will change over time, with modifications to existing analyses and the addition of new metrics and methods. Indeed the validation presented in this paper is only a foundation and has many avenues for expansion. It is therefore vital that code is written in such as way as to make changes and expansions by a community of contributors as easy as possible: COAsT uses a public and open source license; well written documentation and user tutorials; and the documentation contains guidelines for programming practices for contributors. However, having a mechanism for unit testing old and new contributions (with good coverage) is fundamental to maintaining a working codebase. A system of testing is an essential tool for rapid error trapping, when creating a robust codebase by multiple authors, against a backdrop of evolving module dependencies. Suitable coding structures can also assist, for example COAsT makes use of object orientated coding structures, which facilitates independent contributions.

5. **Objective scoring**

   Deriving metrics which minimize room for interpretation is essential when comparing large numbers of model configurations. Where possible, a validation should not be done subjectively, for example visual comparisons of time series. Instead, well defined scoring metrics should be used. This might include commonly used integrated metrics such as the Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and correlation or more complicated probabilistic metrics such as the Continuous Ranked Probability Score (CRPS) (Matheson and Winkler, 1976). Some types of analyses also require different approaches such as extreme value analyses (e.g event based hazard validation).

In the following sections we describe some of the decisions made in the initialisation of our validation workflow. In Section-~~??) . We then use this object interactively with the *Profile* and *Tidegauge* classes to do a comparison between the model and observed data. See Section-2.2 for~~ 2 we provide more information on the ~~observation data used .~~ COAsT package alongside details on the model and observed data used in this paper. In Sections 3 – 4, we showcase some of the analyses available within the package and present results for tidal data, non-tidal residuals and tracers (temperature and salinity) respectively. Finally, in Section-5, we discuss the future of this workflow and opportunities for its expansion.

1.3 ~~Demonstrating the workflow~~

2 Workflow Demonstration: Data

2.0.1 ~~Model data~~

2.1 Model data

We showcase our validation workflow using two different model configurations. Both are built using the NEMO model framework (Nucleus for European Modelling of the Ocean) but at versions 3.6 and 4.0.4 respectively (Madec and Team, 2016, 2019). They belong to the Coastal Ocean (CO) model series, which are used operationally by the UK Met Office for the Northwest European Shelf. At the time of writing, CO7 is the most up to date published version of the configuration (Graham et al., 2017), with CO9 being under development[4]. In this paper, we provide a validation and intercomparison of the CO7 configuration and what we will call CO9p0 configuration. CO9p0 is the first configuration iteration during the development of what will be called CO9. It is intended to perform very similarly to CO7; capturing the change from NEMO v3.6 to NEMO v4.0 with as few other changes as possible. ~~The modeldetails are~~ However differences do arise from structural changes introduced into the model. Known differences include: 1) Bulk forcing implementation: We used the "NCAR" algorithm in CO9p0 in an attempt to closely match the CORE bulk forcing algorithm in CO7 (Large and Yeager, 2009); 2) Lateral diffusion of tracers: We attempt to replicate the constant value used in CO7; 3) Lateral diffusion of momentum: This *is different* between model runs. For stability purposes we deviate from a constant value (as used in CO7) and use the NEMO4 option that varies diffusion according to grid scaling and local velocity; 4) Tracer advection: In CO7 the Total Variance Dissipation (TVD) was used. In NEMO4 the closest equivalent is the Flux Corrected Transport scheme (FCT), which is set to 2nd order in horizontal and vertical directions. 5) Initial conditions differ. The grid and external forcings, as summarised in Table-3 ~~.~~ are the same.

2.1.1 ~~Observation Data~~

2.2 Observation Data

The validation workflow presented below uses in-situ observations of sea surface height, temperature and salinity. For validation of tides and non-tidal sea surface height, we use time series and harmonic analyses from tide gauges around the model domain. For validation of tides specifically, tidal amplitudes and phases are derived from time series taken from multiple sources~~. These locations are shown in Figure-1 and this is discussed further~~ , and some of the raw timeseries data are no longer available. These eclectic sources include tide gauges, moorings and bottom pressure sensors with a very large range in analysis length. Using such a variable dataset allows us to test the harmonic matching and uncertainty discussed in Section-3. To validate high frequency sea surface height time series, we use quality controlled tide gauge data from the GESLA database (Woodworth et al., 2016). The locations for these quality controlled locations are shown in Figure-2. For validation of temperature and salinity, vertical profiles from the EN4 database (Good et al., 2013) are used.

[4]CO8 was skipped for reasons of consistency.

Table 3. Summary of Model configuration and forcing data for CO7 and CO9p0

Bathymetry	EMODnet, September 2015 release
Lateral Boundary Conditions	3D T and S, barotropic velocities and external SSH + tidal forcing.
Atmospheric Forcing	ERA-Interim (Dee et al., 2011)
Atlantic Boundary data	$1/4^{\circ}$ Global Seasonal Forecast System (GLOSEA) version 5 (MacLachlan et al., 2015)
Baltic Boundary data at 12° E	From $1/60^{\circ}$ GETM model of North Sea and Baltic (Gräwe et al., 2015)
Tidal Forcing	TPX07.2 (Egbert and Erofeeva, 2002)
Grid	AMM15 (1.5km) C-grid with rotated pole as described in Graham et al. (2017)
Initial Condition CO7	Initialised at rest on January 1st 1985, using T and S from ORCA025 configuration (Megann et al., 2014).
Initial Condition CO9p0	Started from CO7 restart file from January 1990. Analysis period starts 2004
Vertical Coordinates	51 vanishing quasi sigma levels as in Siddorn and Furner (2013)
Riverine Forcing	Daily climatology of gauge data averaged for 1980–2014 UK data
Output frequency	Daily mean 3D temperature and salinity fields. Hourly 2D sea surface height

3 Workflow Demonstration: Validation against tide gauge data

The high frequency measurements made by tide gauges are useful for validating modelled sea level processes such as tides and non-tidal residuals. In shorter term model simulations, simulating these phenomena accurately can have significance for forecasting (e.g. coastal sea levels) and oceanographic processes (e.g. internal waves, enhanced mixing). On longer time scales the data can be filtered to obtain longer period signals such as trends due to sea level rise. In this section, we use the *Tidegauge* class within the COAsT framework to demonstrate validation of tides and non-tidal residuals. We can use this class to represent and quickly manipulate time series data across multiple locations, which lie along the same time coordinate. COAsT also contains provision for extracting the nearest time series from 2-dimensional model SSH data for each tide gauge location. Using this method provides two instances of the *Tidegauge* class: one for the observed time series and one for the extracted modelled time series. These can be easily compared and analysed.

For high frequency data, the estimation of the tides is a vital step for validation of sea surface height (SSH) in our regional models. In both the observations and the model, these periodic oscillations can both be validated and removed to allow for an analysis of residual signals (non-tidal residuals). Tidal signals exist across a large spectrum of frequencies. Though the most dominant periods are diurnal or semi-diurnal, many shorter or longer periods also exist. These signals of differing periods are known as tidal constituents and are the result of different tidal forcing and periodic interactions. Non-tidal residual signals can be generated by many processes but in coastal regions the ~~modest significant (in terms of coastal hazard)~~ most significant are generated by atmospheric processes such as atmospheric pressure gradients~~and~~, wind generated currents and waves (Vogel, 2015).

To validate and remove the tides, we use a harmonic analysis to estimate tidal amplitudes and phases. This method aims to decompose the sea level signal into a sum of sinusoidal harmonic constituents, with differing frequencies, amplitudes and phases (Vogel, 2015). By determining constituents with predefined frequencies, amplitudes and phases can be estimated using a harmonic analysis of SSH time series. A least squares optimisation is commonly used to do this (see also Fourier methods), where a set of amplitudes and phases are obtained which together form a best fit to the data. This methods also allows for each constituent to be multiplied by a time-varying nodal correction, which is important for extrapolating tidal predictions.

3.1 Matched harmonic analysis (MHA)

When both validating the tidal signal and removing it from the total signal, it is important that the harmonic analysis of modelled and observed data is equivalent. Each set of harmonics may be calculated from time series of different lengths, time periods and using different constituent sets. Furthermore, differences in software methods can add more variance in the output, creating some uncertainty in any comparisons between the model and observations. These variations can cause differences in the amplitudes and phases of estimated harmonics, leading to difficulties comparing harmonic information from model and observations. Similarly, these harmonic differences may propogate into timeseries obtained through the subtraction of tides (e.g. non-tidal residuals for storm surge validation).

Below, we outline the steps used in our validation workflow to ensure that analyses are equivalent. ~~We~~ When these are true, we call this a Matched Harmonic Analysis (MHA).

1. Obtain observed time series and extract model SSH time series at the nearest model grid cell.
2. Subsample or interpolate both time series to be on the same frequency – for example hourly.
3. Where there are missing or flagged data in the observations, also remove these data from the model time series.
4. Ensure both time series start and end at the same time.
5. Perform a harmonic analysis of each time series, using the same constituent set. The constituent set should be determined using the Rayleigh Criterion (Vogel, 2015).
6. For each time series, reconstruct a tidal signal from the estimated tidal harmonics. Subtract these signals from each time series to obtain non-tidal residuals.

By not following these steps, significant uncertainty may be introduced into any comparisons between modelled and observed harmonics. This uncertainty will also be propagated into comparisons of non-tidal residuals. At the time of writing, COAsT contains a number of routines which make applying the MHA to modelled and observed data quick and easy. By representing both observed and modelled timeseries in an instance of the *Tidegauge* class, we can quickly slice out equivalent time periods, interpolate to the same frequency and match missing values across each dataset.

3.1.1 Harmonic uncertainty estimation where MHA is not possible

In some cases, it may not be possible to follow all of the steps above, e.g. when time series are short or time periods do not coincide. For these cases, we instead attempt to quantify and apply this uncertainty. This is true for the large dataset of harmonics we use for tidal validation (described in Section-2.2). To estimate uncertainty in observed harmonics, we use modelled SSH time series on an hourly frequency. We have extracted these time series at the nearest model points to each location in the observed dataset described in Section-2.2. For each observation location the analysis lengths have been identified from observations and these are shown in Figure-3(a). For each model SSH time series, an ensemble of harmonic analyses is performed. Each member of the ensemble contains a harmonic analysis with a length defined by the analysis length of the corresponding observation and an appropriate constituent set according to the Rayleigh Criterion. The ensemble standard deviation is then calculated at each location for each constituent and used to define the observation uncertainty. From an initial set of analyses, analysis length and the time period of analysis were found to be two of the most influential generators of uncertainty. When isolated, the chosen constituent set was also found to have an effect, although not as large.

Results from this analysis are shown for M2 in Figure-3(b)-(c). For amplitude, they are expressed as a percentage of the amplitude obtained from a 10-year harmonic analysis. The standard deviations in amplitude can reach over 3% at some ports, which can be large at locations where M2 is also large (e.g. the Severn Estuary). Differences in phase appear less consequential, reaching a few degrees. This analysis shows that there is a significant degree of variability in harmonic analyses when time series of differing length and period are used. In turn, this variability can become uncertainty when comparing harmonics from model and observed data. Using an MHA can reduce this uncertainty, but where this is not possible ~~it must not be ignored~~the uncertainty must be considered instead. In the next section we show an analysis using both the MHA approach and an application of the harmonic variability estimated here.

3.2 Validation of tides

As discussed above, we validate tides using a comparison of tidal amplitudes and phases obtained using a harmonic analysis. The harmonic analysis is obtained using hourly SSH data for the full duration of the model run. SSH time series are extracted from the model at the nearest wet grid point for this purpose. Observations used are described in Section-2.2.

Maps of errors in amplitude and phase for six of the largest semidiurnal and diurnal constituents are shown in Figures 1 ~~-~~and 4. The raw errors are shown in the left two columns. The rightmost columns show the difference between absolute errors for each of the model configurations. As discussed in Section-3.1, we ~~cannot~~ would ideally apply a matched harmonic analysis ~~to this analysis as many of the harmonics are derived using data that falls~~ between the model and observations. However, some historical data are only available as pre-computed harmonics, or the observations fall outside of the ~~model run window. Instead we apply the estimated harmonic uncertainties to~~ simulation window. In these circumstances we can compute an estimated harmonic uncertainty for any comparison between ~~the model and the observations~~modelled and observationally derived harmonics (as discussed in Section-3.1.1): Where differences between the model and the observations are smaller than the uncertainty, they are deemed insignificant and coloured grey in the figures. This is done independently for each harmonic

constituent. By doing this, we mask out points where the model is imperceptibly close to the observations and, most importantly, make no judgements or comparisons on the two models performance at these points.

In the figure, we see that across all constituents and for both amplitude and phase, the spatial distribution of errors between the two models is structurally similar. Visually, the two models look very close, and this is reinforced somewhat by the improvement panels. The errors at many points, especially those away from the coast, are similar enough that they lie within the observational uncertainty. Where there are significant differences, they are small – being on the order of a few centimetres or degrees. The models are closest for K1 amplitude and phase, where there are very few significant differences. Notably, P1 shows more significant differences than the other constituents, especially for phase.

3.3 Validation for non-tidal residuals

Whereas harmonics are useful for validating periodic signals due to tides, they do not capture other motions, such as those caused by interactions with the atmosphere. Storm surges are the most impactful of these in coastal regions, and can be measured with metrics such as the non-tidal residual and skew surge. Therefore, validation of the full SSH signal, and its components in the time domain, is important. In this section we consider the full SSH signal as well as a validation of non-tidal residuals. Here, we again use the functionality within the *Gridded* and *Tidegauge* classes of the COAsT package. In particular, we are able to use these classes to extract the nearest time series from the *Gridded* object, subtract tides from the signal, calculate some simple metrics and perform a basic extreme value analysis.

As discussed above, to derive non-tidal residuals (NTR) a tidal signal must be subtracted from the full SSH signal. In operational scenarios, a separate tide-only run may be performed to get a time series of tides at all model locations, which can be used to estimate non-tidal residuals. However, interactions between the tidal and non-tidal components (for example see Prandle and Wolf (1978)) mean that this may not be an accurate way of removing the tide and is likely to leave a significant periodic signal in the resulting residual. Therefore, it is recommended that a harmonic analysis of the full signal is performed and the reconstructed water levels subtracted. To avoid uncertainty in tidal harmonics ~~propogating~~ propagating into the residual signal, we ~~apply~~ only use data for which the matched harmonic analysis (described in Section-3.1) can be applied.

Once non-tidal residuals have been derived, various statistics can be estimated. Errors and absolute errors (or MAE/RMSE) can be sensitive to residual tidal constituents that may not have been completely removed by the harmonic analysis process. This is especially true when non-tidal residuals are small such as during calm atmospheric conditions like a high pressure system. In addition, it is often the largest non-tidal residuals which we are most interested in modelling accurately, as they have the largest impact. The COAsT package currently has a number of options available for extreme value analysis of non-tidal residuals. As a part of the validation workflow, the package will calculate:

1. Daily and Monthly maxima at all tide gauge locations. Count the number of each over specified thresholds
2. Identify independent peaks in the signal and count those over specified thresholds.
3. Integrate the total time spent by a signal over specified thresholds.

This is a simple extreme value analysis and can be expanded upon in future. For example, the application of a generalized extreme value distribution to signal peaks or the generalized pareto distribution to daily or monthly maxima.

Figure-2 shows the error in the standard deviation of total water level for each of the two models along with the difference in errors between the two models. Where positive, this difference indicates that CO9p0 performed better (i.e. the absolute standard deviation errors were smaller). This metric can be used as a proxy for the atmospherically influenced tidal range tidal range, averaged over time. It is a good way to measure the error across all harmonic constituents. Both models have ~~are~~ similar spatially, overestimating standard deviations at most locations. The most notable difference between the two models is in the Severn Estuary, where tidal ranges are large. Here, the CO9p0 configuration performs much better. This is likely because of the parallel improvements seen in the amplitude of the largest constituents (see Figure-1).

Figure-5 shows the correlations between modelled and observed NTR time series over the full duration of the run. This gives us information about how well the modelled and observed signals move together, as well as the timing of the non-tidal residuals. In this case, the CO9p0 model performs better at the majority of locations, especially those in estuarine areas such as the Severn and Thames estuaries.

Figure-6 shows two examples of extreme value statistics for the non-tidal residuals, expressed as a function of thresholds. Here, non-tidal residual thresholds have been defined between $0\,\mathrm{m}$ and $1.5\,\mathrm{m}$. Figure-6(a) shows the number of modelled independent peaks over each threshold divided by the number of observed peaks. For example, a value of 0.5 indicates that the model generated only a half of the number of peaks over a given threshold as were observed. Peaks were defined as independent maxima, separated by at least 12 hours, to avoid double counting the same events. Figure-6(b) shows the total time spent over each threshold, again as a proportion of that in the observations. Together, these two figures show us how well the models are able to capture large events such as storm surges over the period of the model run. However the figures also show that both configurations are underestimating large events, and the larger the observed event gets, the more their number is underestimated. It also seems that neither model is sufficiently sensitive to atmospheric forcing and that coastal effects, such as resonance, are not being adequately represented.

4 Workflow Demonstration: Validation against profile data

To validate temperature and salinity in the model configurations, we compare them to in-situ profiles from the EN4 database (Good et al., 2013). As discussed, the *Profile* object within the COAsT framework is well suited to handle this type of data. The routines within it allow us to read from common profile databases such as EN4 and World Ocean Database, represent multiple profiles in a single standardised object, manipulate this data in time and space and extract comparable model profiles. Similarly to the analysis of tide gauge data above, the extracted model data may also be represented using a *Profile* object for quick and easy comparisons.

Model profiles are extracted using multiple independent interpolations, both in space and time. The horizontal and vertical interpolations are also treated independently. Horizontal interpolations can be done quickly using either a nearest neighbour or bilinear approach. For the analyses below, we have used a nearest neighbour interpolation, taking the nearest model wet

point to each observation. However, in some cases this may result in a model cell being chosen that is too far from the observation location, such as near complex coastlines. These points are few, but to avoid them having an effect on the analysis, all interpolated points further than $5\,\mathrm{km}$ from from the nearest observation location are omitted. This horizontal interpolation results in a time series of data at every model depth. This is then interpolated onto the measurement time of the nearest profile using a linear interpolation.

At this point we have a model profile for each observed profile, however the data still lies on model depth levels so a vertical interpolation is still required. Since this is for comparison purposes, it is useful to interpolate both the model and observations onto the same reference depths at each location. This can be done in a single interpolation step, i.e. interpolating directly onto a set of reference depths, however this can cause problems where the vertical coordinate density varies drastically between the two. Errors in the interpolation may be falsely interpreted as errors in the model. This is likely to be most prominent in areas with high vertical gradients in variables like density, for example near a pycnocline. We can reduce this problem by doing the vertical interpolation in two steps:

1. Interpolate the model onto observation depths if the latter are sparser, or vice versa.

2. Interpolate both the observations and interpolated model data onto the same reference depths.

The COAsT package is able to interpolate sets of profiles onto reference depths, or onto the depths of another *Profile* object. This allows for flexibility with vertical interpolation and applying the two step process discussed above.

Now that we have a set of modelled and observed profiles on the same reference depths, we can do a comparison. At higher resolutions, where smaller features may be resolved, traditional metrics such as the Mean Absolute Error and Root Mean Squared Error may not be the best options for validation. These metrics can result in a double counting effect for small scale features such as eddies. For example, a higher resolution model may resolve a certain scale of eddy when a lower resolution model may not. However, if an eddy is not at the right place at the right time, then MAE will count both the error at the observed eddy location and at the modelled eddy location. To counter this, a probabilistic metric should be used instead. To do this, we have used the Continuous Ranked Probability Score (CRPS) (Matheson and Winkler, 1976) at the ocean surface. This is generally used with an ensemble of model states to compare the Cumulative Distribution Functions (across ensembles) of the model and a single observation at a specific location. However, when a small enough neighbourhood around a location is taken, it can also be considered sufficiently random and used in place of the ensemble. This is the formulation used here and we apply it to both surface temperature and salinity:

$$CRPS(F,x) = \int_{-\infty}^{\infty} (F(y) - \mathbb{1}(y-x))^2 dy, \tag{1}$$

where F is a cumulative distribution function (CDF), x is a single observation and $\mathbb{1}$ is a Heaviside function that is equal to 1 where its argument is greater than or equal to zero, and 0 otherwise. The CDF F is ~~comprised of~~ derived from all model values from within some predefined radius around the observation x, as described above. $F(y)$ is then the value of this CDF

for a single element of the model radial dataset. More intuitively, the CRPS can be thought of as the mean square error between modelled and observed CDFs.

In order to identify regional differences in model errors, we average estimated errors into regional and depth bins. The regions used in this paper are shown in Figure-7. The validation workflow within COAsT includes flexible routines within the *mask_maker* class for averaging errors both into regional areas and into depth bins. This is a useful method for obtaining averaged objective error metrics whilst still targeting specific areas of the domain. Figures 8 – 9 show the mean absolute errors in depth levels for levels less than $150\,\mathrm{m}$. The absolute error metric allows us to see how each model performs~~in each region,~~ , for temperature and salinity, at different depths and in each region. In addition, these plots are split into summer (JJA) and winter (DJF) seasons. We can see that both of the model configurations studied in this paper are similar at all depths and most regions. Figures 10-11 show CRPS values for SST and SSS respectively. In this case, we see similar CRPS values across all radii. For SST, the CO7 configuration performs best (i.e. smallest values) for most regions. The opposite is true for SSS. For both variables, the Norwegian Trench and Kattegat generally have the highest (worst) CRPS values.

5 Further Discussion

In this paper, we have presented the underlying principles used to develop a new standardised workflow for validating model data. The principles introduced ~~are ensuring~~ aim to ensure that assessments are scaleable with data size, independent of data source, reproducible, have a codebase that is easily expandable and use metrics which provide objective comparisons. They aim to ensure that any analysis of model data can be easily applied, cited, reproduced and interpreted. Alongside our discussion, we showcased an existing workflow for validating modelled sea surface height, temperature and salinity against tide gauge and profile observations. In the examples shown, we have adhered to ~~all of~~ the principles set out in this paper. We used the COAsT Python package, which offers a set of standardised oceanographic data structures which are ideal for comparing model data with observation. The package~~is an important tool~~, along with its components (especially Xarray), are important tools, allowing us to adhere to the assessment principles with ease.

This kind of validation framework can be used as an integral part of the model development process. Such a process is often iterative, with the tuning of various processes and parameters being necessary. This kind of workflow, again with the added benefits of the standardised COAsT framework, can be run quickly and with minimal changes for each new set of model output. In addition, it is recommended that model configurations are evaluated and tested using the same validation and the same sets of metrics for consistency. The version control and DOI system used by COAsT also means that any validation code may be reverted back to in the future if necessary. For example, this may be needed if there are differences or errors in output that need to be evaluated.

The assessment metrics presented in this paper are by no means exhaustive, and serve as a demonstration and foundation for future additions. As discussed throughout this paper, a real advantage of our approach is that new assessments may be added to the analysis in an easy, modular fashion, whilst preserving the reproducibility of past iterations of the code. New validations

may be designed to incorporate more sources of observations, new metrics and diagnostics, new variables and comparisons to other models.

The philosophy introduced in this paper has been used for our own development with great success. We present these reflections in hope that they are useful to the wider community.

Author Contributions

DB wrote the primary manuscript draft with additional contributions from JP, EOD and JW. Experiment design was lead by DB, with additional contributions from JP, EOD and JW. Development of model configurations was lead by EOD, with contributions from DB and JP.

Acknowledgements. This work was conducted through the Weather and Climate Science for Service Partnership (WCSSP) India, a collaborative initiative between the Met Office, supported by the UK Government's Newton Fund, and the Indian Ministry of Earth Sciences (MoES).

Appendix: Code Availability Statement

The code used in this paper is openly available as part of the COAsT toolbox. This can be accessed via Github: https://github.com/British-Oceanographic-Data-Centre/COAsT. At the time of writing, the most up to date version information is available on Zenodo doi: https://doi.org/10.5281/zenodo.6778993. Additionally, scripts that used COAsT for analysis and to create figures are available in a separate repository here: https://github.com/JMMP-Group/NEMO_validation.

References

Dee, D. P., Uppala, S. M., Simmons, A. J., Berrisford, P., Poli, P., Kobayashi, S., Andrae, U., Balmaseda, M. A., Balsamo, G., Bauer, P., Bechtold, P., Beljaars, A. C. M., van de Berg, L., Bidlot, J., Bormann, N., Delsol, C., Dragani, R., Fuentes, M., Geer, A. J., Haimberger, L., Healy, S. B., Hersbach, H., Hólm, E. V., Isaksen, L., Kållberg, P., Köhler, M., Matricardi, M., McNally, A. P., Monge-Sanz, B. M., Morcrette, J.-J., Park, B.-K., Peubey, C., de Rosnay, P., Tavolato, C., Thépaut, J.-N., and Vitart, F.: The ERA-Interim reanalysis: configuration and performance of the data assimilation system, Quarterly Journal of the Royal Meteorological Society, 137, 553–597, https://doi.org/https://doi.org/10.1002/qj.828, 2011.

Egbert, G. D. and Erofeeva, S. Y.: Efficient Inverse Modeling of Barotropic Ocean Tides, Journal of Atmospheric and Oceanic Technology, 19, 183 – 204, https://doi.org/10.1175/1520-0426(2002)019<0183:EIMOBO>2.0.CO;2, 2002.

Good, S. A., Martin, M. J., and Rayner, N. A.: EN4: Quality controlled ocean temperature and salinity profiles and monthly objective analyses with uncertainty estimates, Journal of Geophysical Research: Oceans, 118, 6704–6716, https://doi.org/https://doi.org/10.1002/2013JC009067, 2013.

Graham, J. A., O'Dea, E., Holt, J. T., Polton, J. A., Hewitt, H. T., Furner, R., Guihou, K., Brereton, A., Arnold, A., Wakelin, S. L., Sánchez, J. M., and Adame, C. G. M.: AMM15: a new high-resolution NEMO configuration for operational simulation of the European north-west shelf, Geoscientific Model Development, 11, 681–696, 2017.

Gräwe, U., Holtermann, P., Klingbeil, K., and Burchard, H.: Advantages of vertically adaptive coordinates in numerical models of stratified shelf seas, Ocean Modelling, 92, 56–68, https://doi.org/https://doi.org/10.1016/j.ocemod.2015.05.008, 2015.

Hoyer, S. and Hamman, J.: xarray: N-D labeled Arrays and Datasets in Python, Journal of Open Research Software, 5, 10, https://doi.org/http://doi.org/10.5334/jors.148, 2017.

Large, W. G. and Yeager, S. G.: The Global Climatology of an Interannually Varying Air–Sea Flux Data Set, Climate Dynamics, 33, 341–364, https://doi.org/10.1007/s00382-008-0441-3, 2009.

MacLachlan, C., Arribas, A., Peterson, K. A., Maidens, A. V., Fereday, D. R., Scaife, A. A., Gordon, M., Vellinga, M., Williams, A. I. L., Comer, R. E., Camp, J., Xavier, P., and Madec, G.: Global Seasonal forecast system version 5 (GloSea5): a high-resolution seasonal forecast system, Quarterly Journal of the Royal Meteorological Society, 141, 2015.

Madec, G. and Team, N. S.: NEMO ocean engine, https://doi.org/10.5281/zenodo.3248739, 2016.

Madec, G. and Team, N. S.: NEMO ocean engine, https://doi.org/10.5281/zenodo.1464816, 2019.

Matheson, J. E. and Winkler, R. L.: Scoring rules for continuous probability distributions., Manage Sci., 22, 1087–1095, 1976.

Megann, A., Storkey, D., Aksenov, Y., Alderson, S., Calvert, D., Graham, T., Hyder, P., Siddorn, J., and Sinha, B.: GO5.0: the joint NERC–Met Office NEMO global ocean model for use in coupled and forced applications, Geoscientific Model Development, 7, 1069–1092, https://doi.org/10.5194/gmd-7-1069-2014, 2014.

Polton, J., Harle, J., Holt, J., Katavouta, A., Partridge, D., Jardine, J., Wakelin, S., Rulent, J., Wise, A., Hutchinson, K., Byrne, D., Bruciaferri, D., O'Dea, E., De Dominicis, M., Mathiot, P., Coward, A., Yool, A., Palmiéri, J., Lessin, G., Mayorga-Adame, C. G., Le Guennec, V., Arnold, A., and Rousset, C.: Reproducible and Relocatable Regional Ocean Modelling: Fundamentals and Practices, Geoscientific Model Development, 16, 1481–1510, https://doi.org/10.5194/gmd-16-1481-2023, 2023.

Prandle, D. and Wolf, J.: The interaction of surge and tide in the North Sea and River Thames, Geophysical Journal International, 55, 203–216, https://doi.org/https://doi.org/10.1111/j.1365-246X.1978.tb04758.x, 1978.

Rocklin, M.: Dask: Parallel computation with blocked algorithms and task scheduling, in: Proceedings of the 14th python in science conference, 130-136, Citeseer, 2015.

Siddorn, J. A. and Furner, R.: An analytical stretching function that combines the best attributes of geopotential and terrain-following vertical coordinates, Ocean Modelling, 66, 1–13, 2013.

Vogel, M.: Sea-level Science: Understanding Tides, Surges, Tsunamis and Mean Sea-level Changes, by David Pugh and Philip Woodworth, Contemporary Physics, 56, 394–394, https://doi.org/10.1080/00107514.2015.1005682, 2015.

Woodworth, P. L., Hunter, J. R., Marcos, M., Caldwell, P., Menéndez, M., and Haigh, I.: Towards a global higher-frequency sea level dataset, Geoscience Data Journal, 3, 50–59, https://doi.org/https://doi.org/10.1002/gdj3.42, 2016.

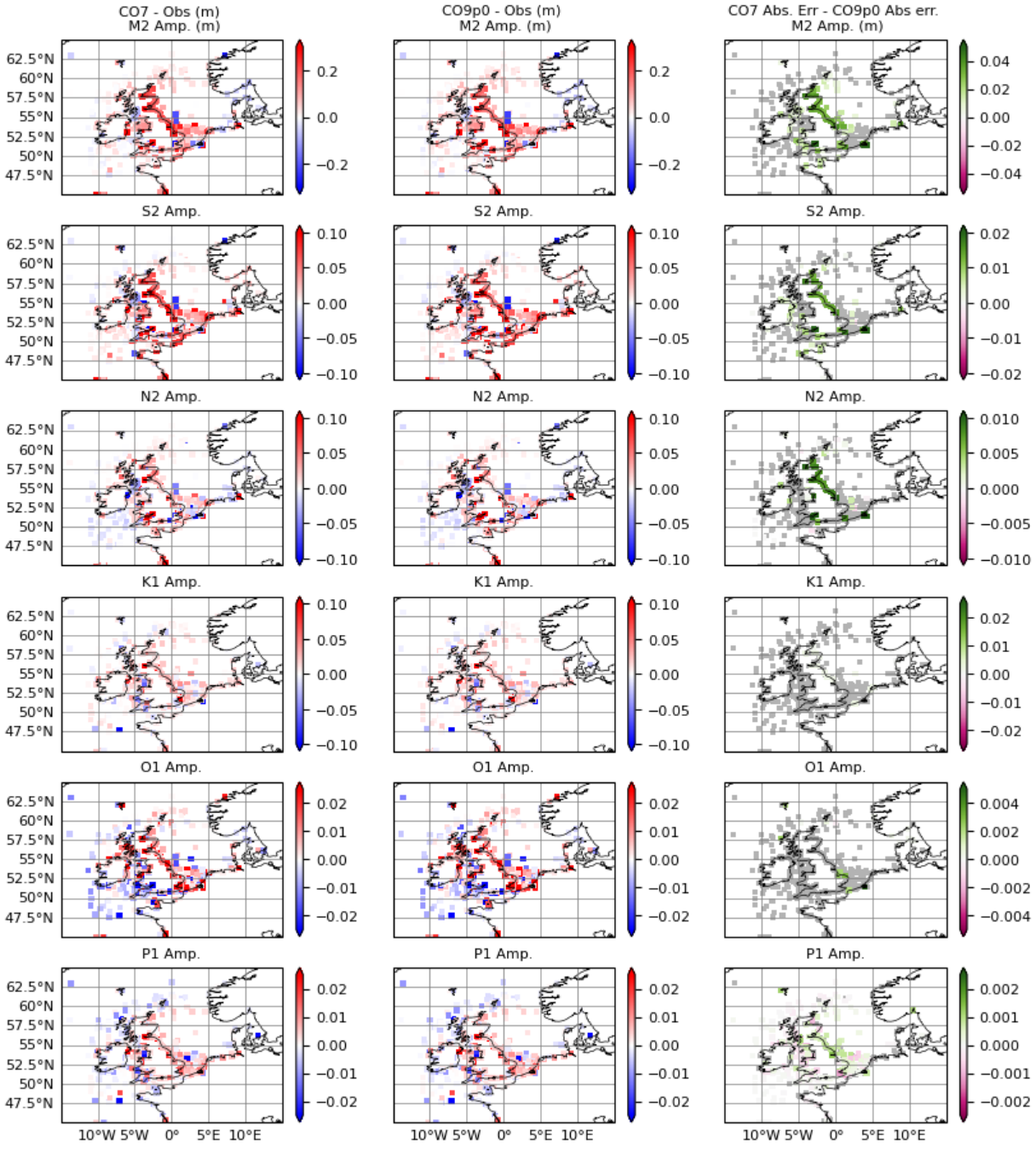

Figure 1. Errors in amplitude (m) for **left:** the CO7 model run, **middle:** the CO9p0 model run. Errors are calculated as model minus observations, meaning positive values indicate overestimation by the model. The rightmost column shows the difference in absolute error between the two model runs, calculated as CO7 - CO9p0. Positive values here indicate an 'improvement' of CO9p0 when compared to CO7 (i.e. a smaller absolute error). Grey values show locations where differences were smaller than the estimated uncertainty in the model-observation comparison.

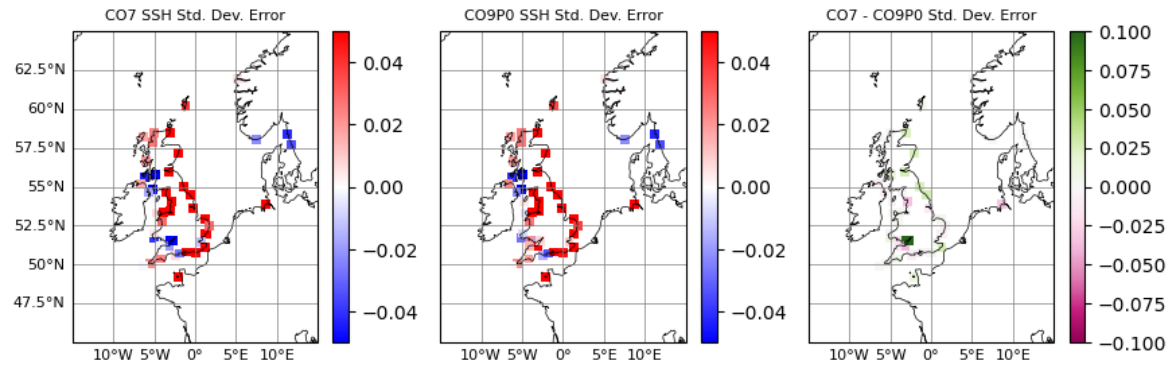

Figure 2. Errors in Total Water Level standard deviations, calculated over the 10 year model run for the CO7 and CO9p0 configurations. The two panels on the left show the errors for the CO7 and CO9p0 configurations.

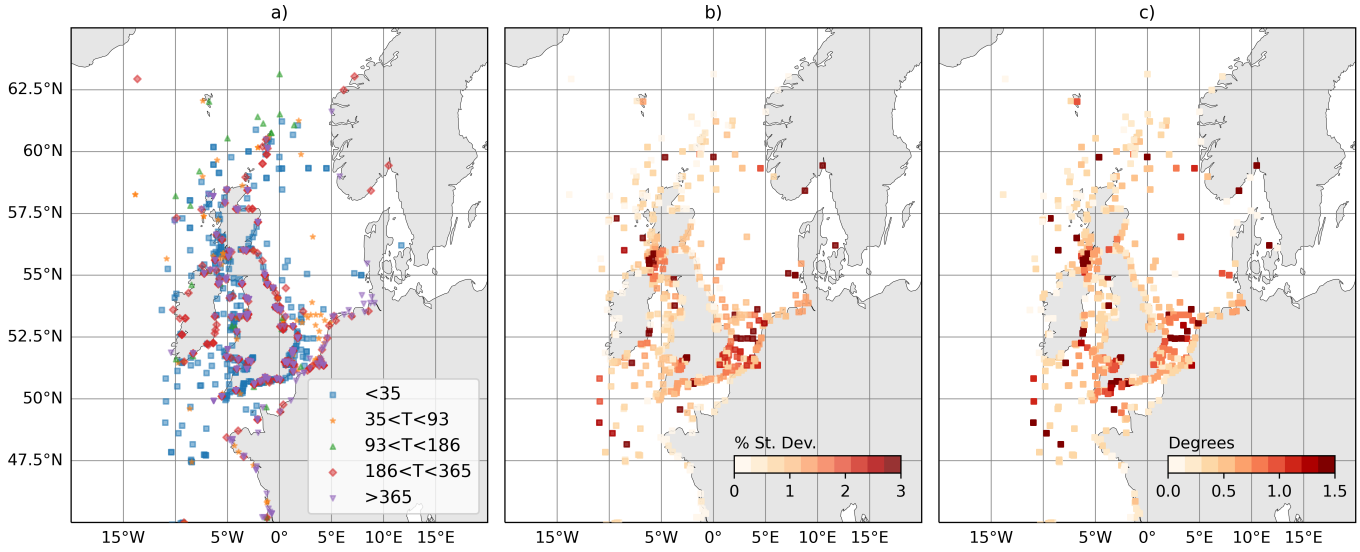

Figure 3. Estimation of uncertainty in M2 amplitude and phase resulting from different analysis lengths and constituent sets. **a)** Analysis lengths used for each observation location. **b)** Ensemble standard deviation in M2 amplitude as a proportion of amplitude (~~metres~~%). **c)** Ensemble standard deviation in M2 phase (degrees).

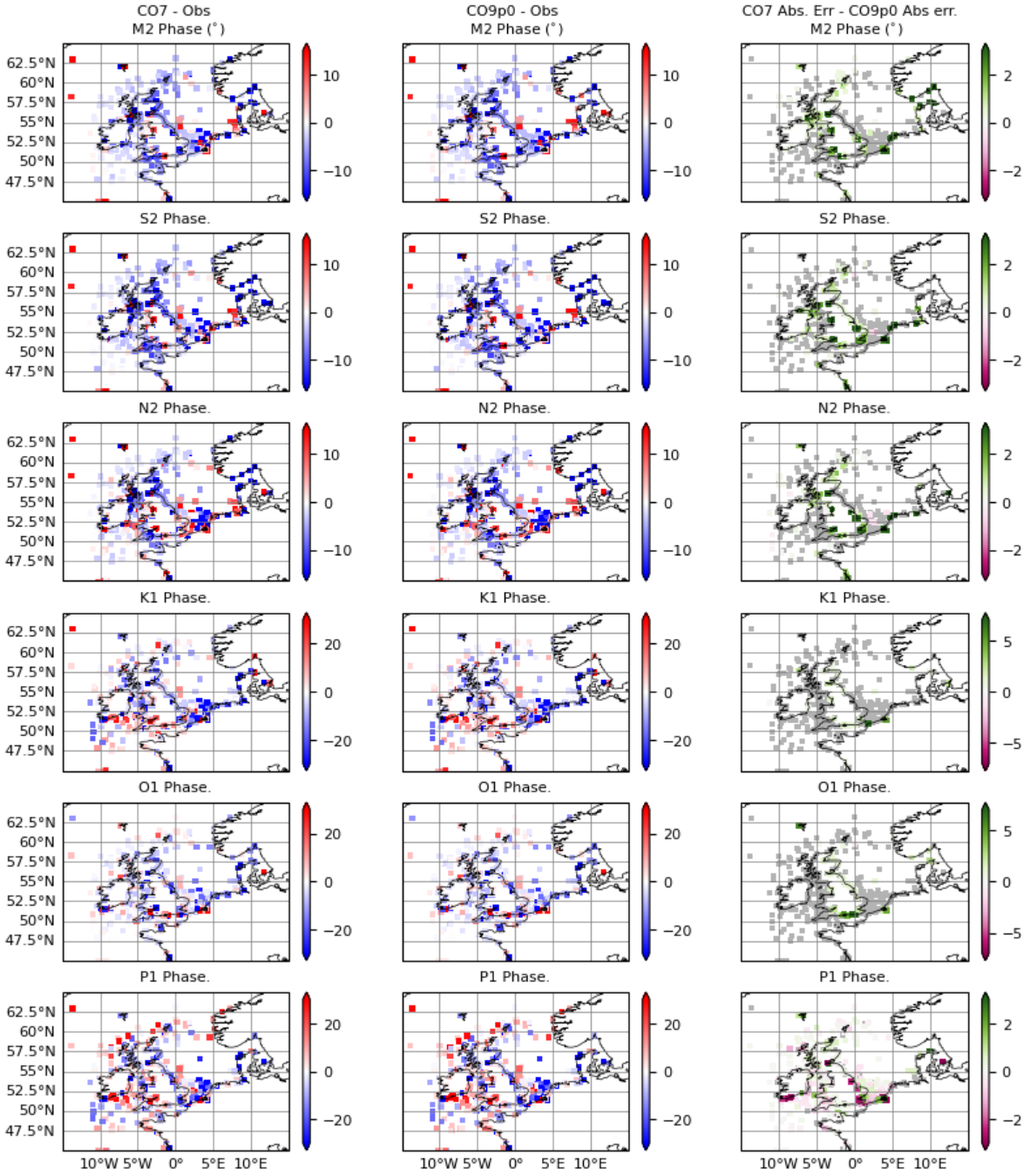

Figure 4. Errors in phase (degrees) for **left:** CO7 model run, **middle:** the CO9p0 model run. Errors are calculated as model minus observations, meaning positive values indicate overestimation by the model. The rightmost column shows the difference in absolute error between the two model runs, calculated as CO7 - CO9p0. Positive values here indicate an 'improvement' of CO9p0 when compared to CO7 (i.e. a smaller absolute error). Grey values show locations where differences were smaller than the estimated uncertainty in the model-observation comparison.

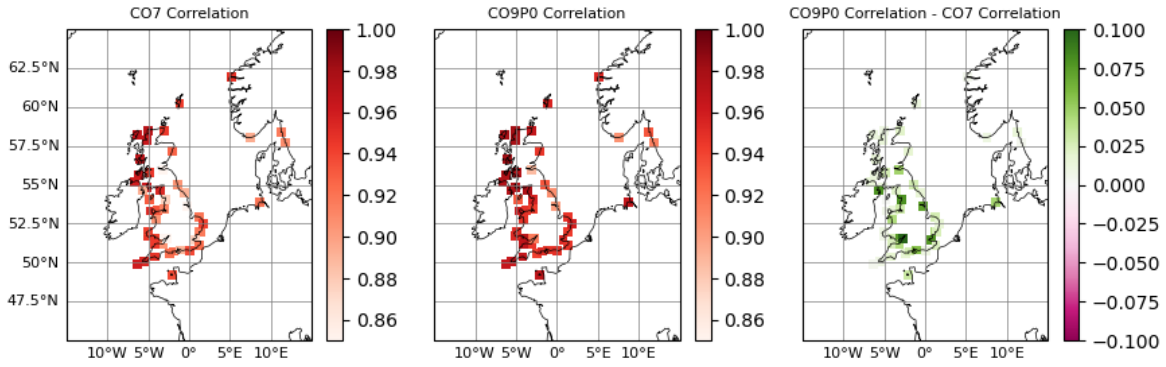

Figure 5. Correlations between modelled and observed non-tidal residuals at tide gauge locations. The left and middle panel show correlations for the CO7 and CO9p0 model runs respectively. The right panel shows (where positive) where the CO9p0 run had higher correlations than the CO7 run.

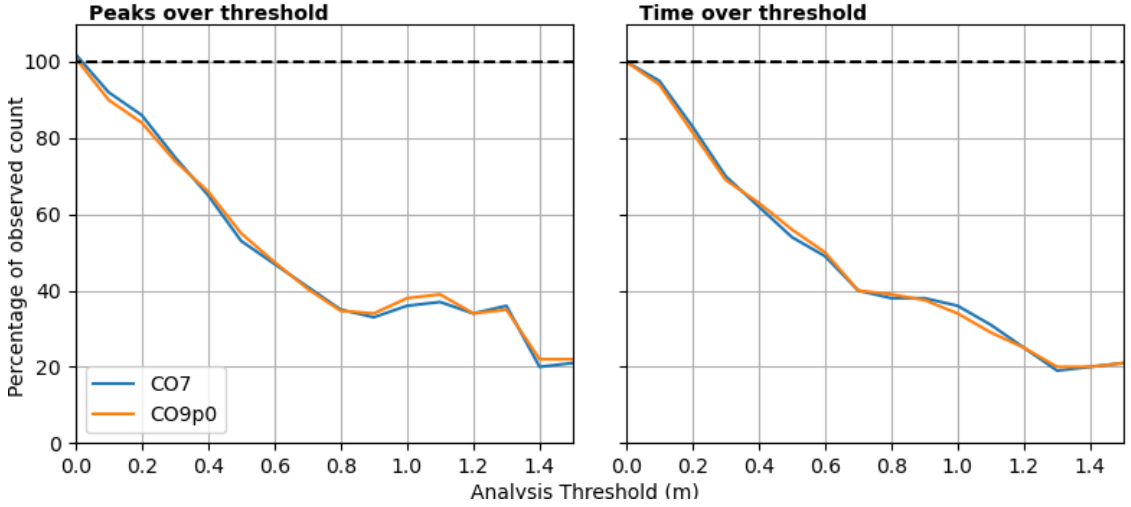

Figure 6. Threshold statistics for non-tidal residuals in the CO7 and CO9p0 model configurations. **Left:** The number of independent peaks over a given threshold, as a proportion of the number of peaks in the observations. **Right:** Total time spent over threshold, as a proportion of the time spent by observations.

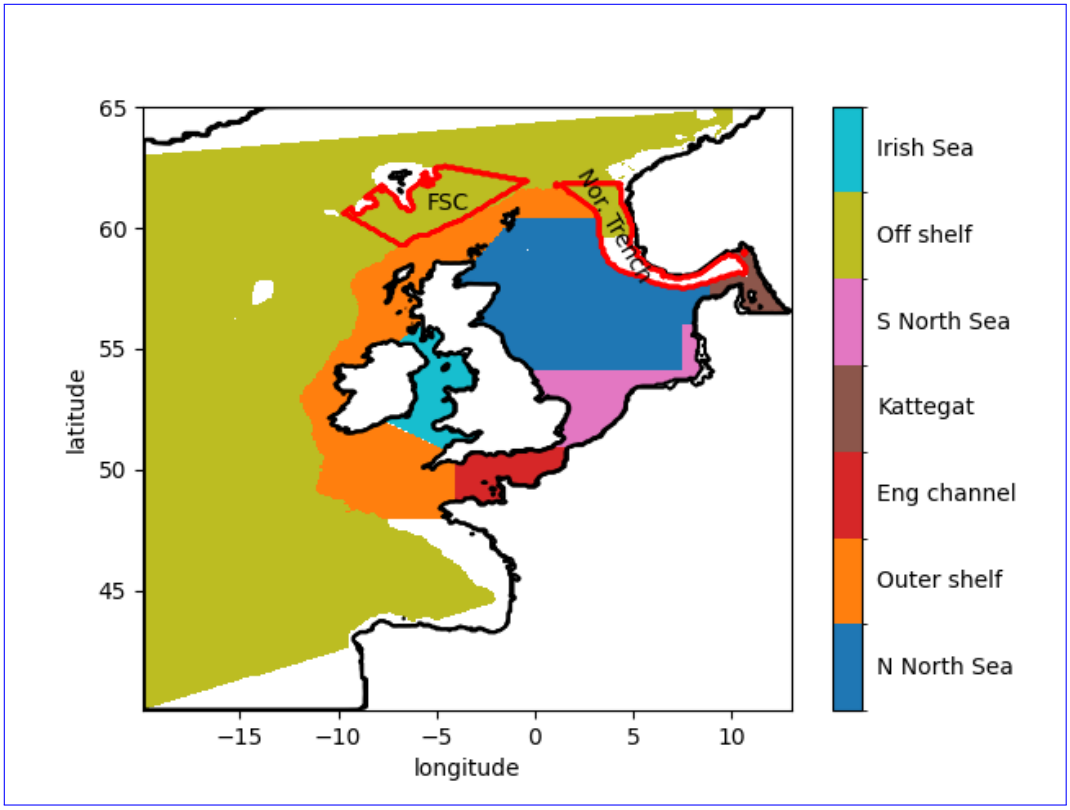

Figure 7. ~~Absolute temperature errors with depth for seven regions in~~ Illustration of the ~~AMM domain. Comparisons made between the CO9p0 and CO7 model runs. The black dashed horizontal lines show the mean bathymetric depth across the profiles~~ different averaging regions used in ~~each region~~this study.

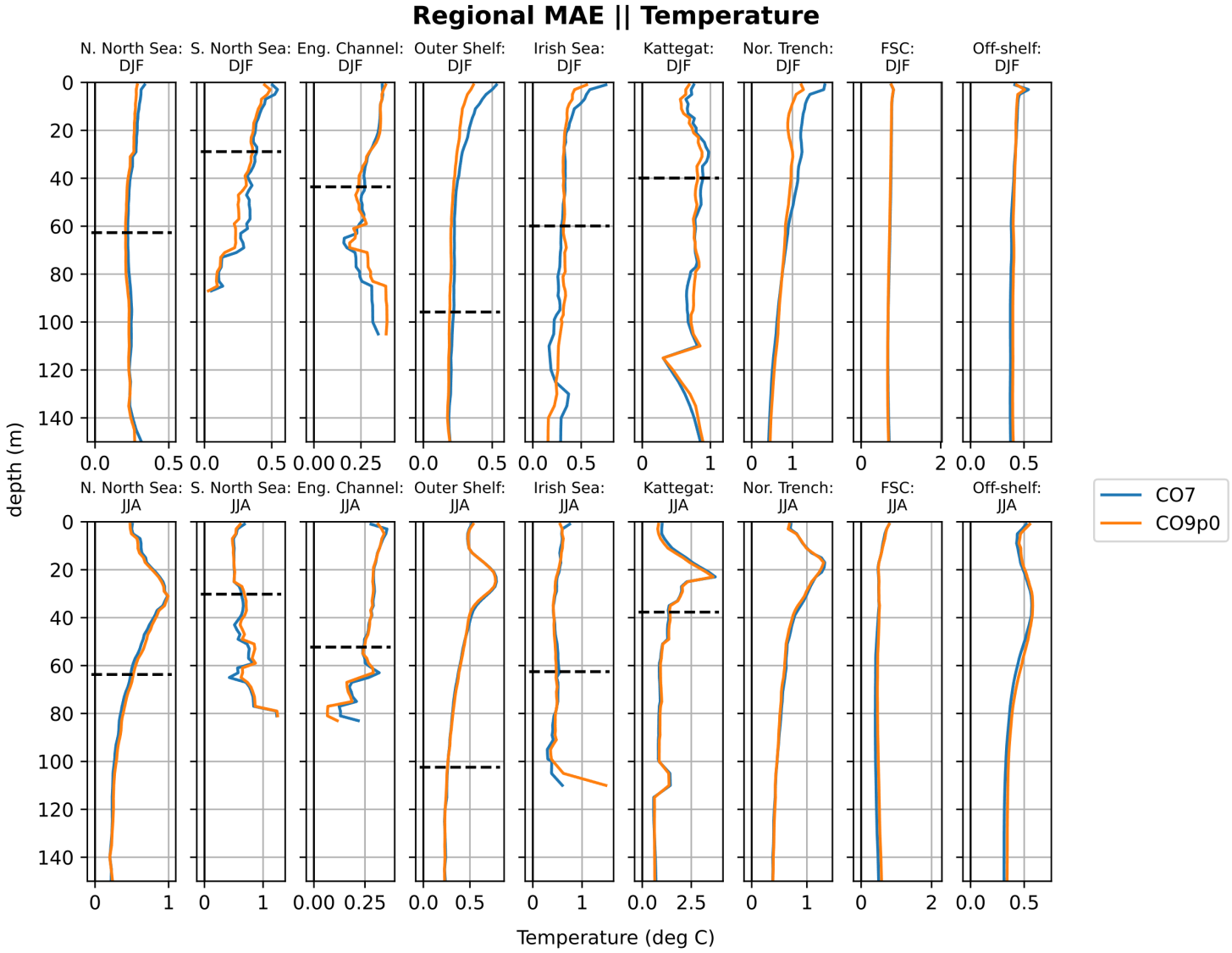

Figure 8. Absolute temperature errors with depth for regions in the AMM domain. Comparisons made between the CO9p0 and CO7 model runs. The black dashed horizontal lines show the mean bathymetric depth across the profiles used in each region (where displayed profile is deep enough).

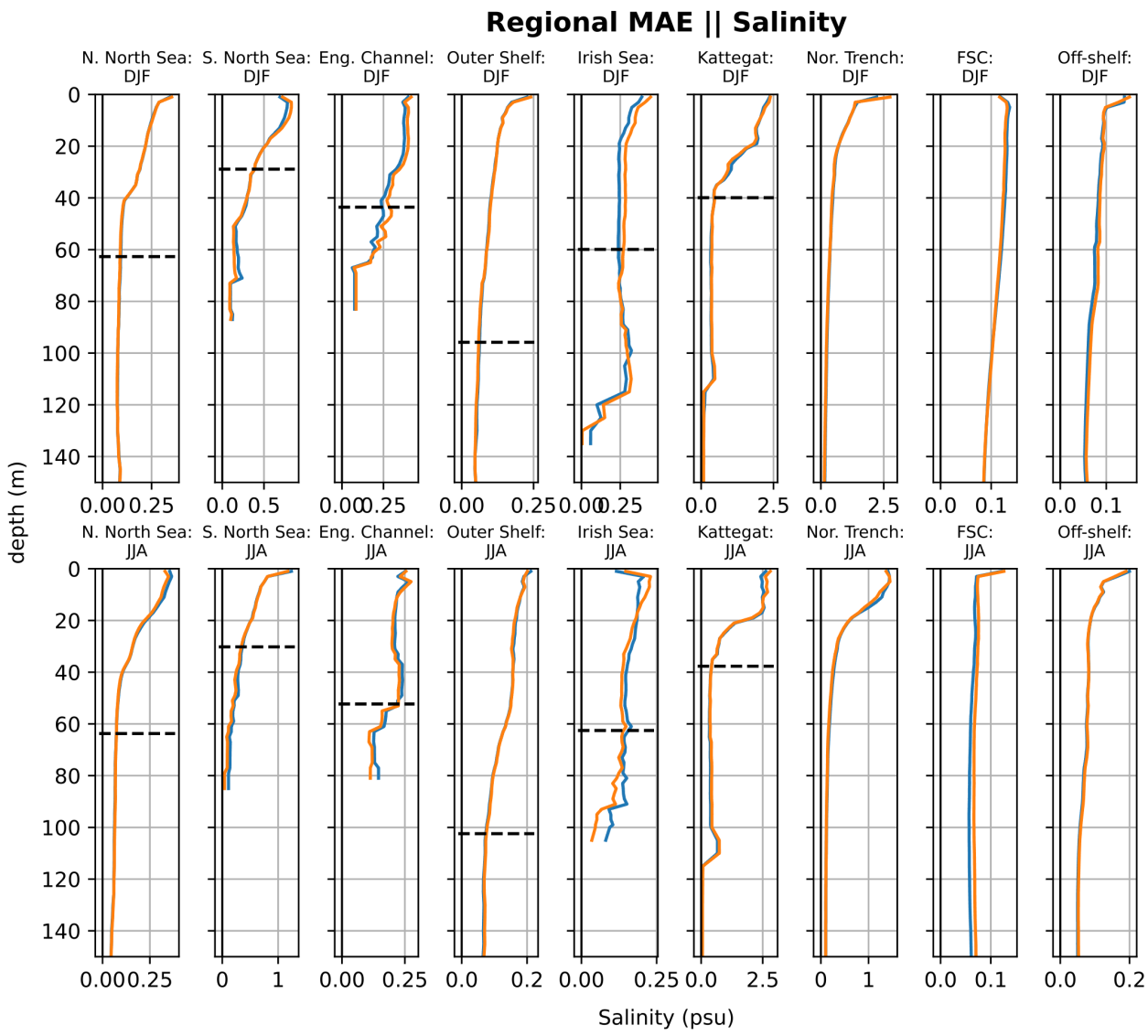

Figure 9. Absolute salinity errors with depth for ~~seven~~ regions in the AMM domain. Comparisons made between the CO9p0 and CO7 model runs. The black horizontal lines shows the mean bathymetric depth across the profiles used in each region (where displayed profile is deep enough).

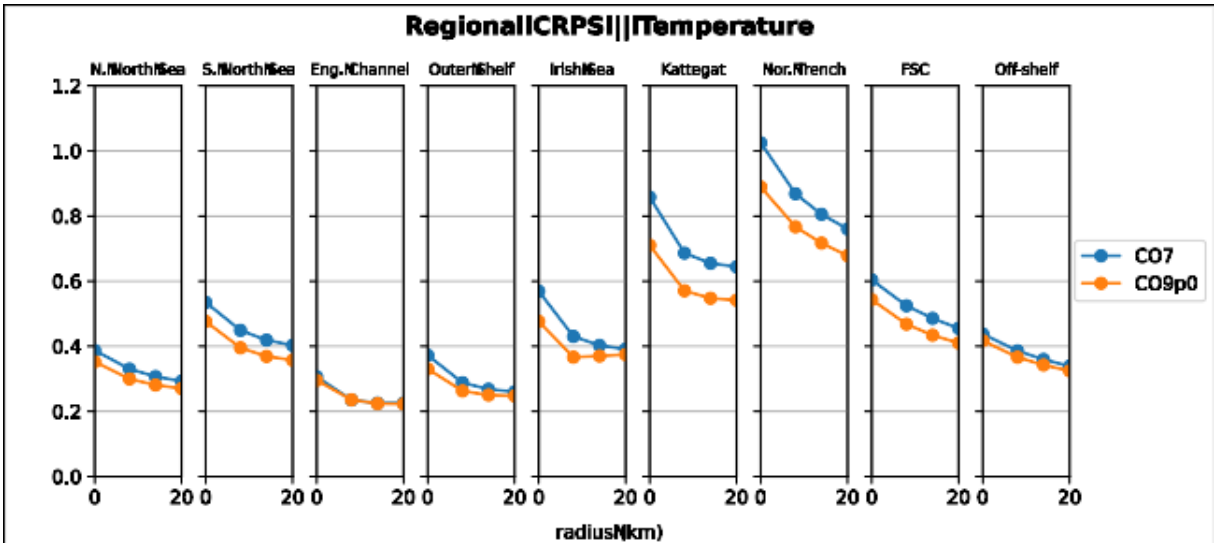

Figure 10. SST Continuous Ranked Probability Score (CRPS) for eight regions in the AMM domain. Comparisons made between the CO9p0 and CO7 model runs.

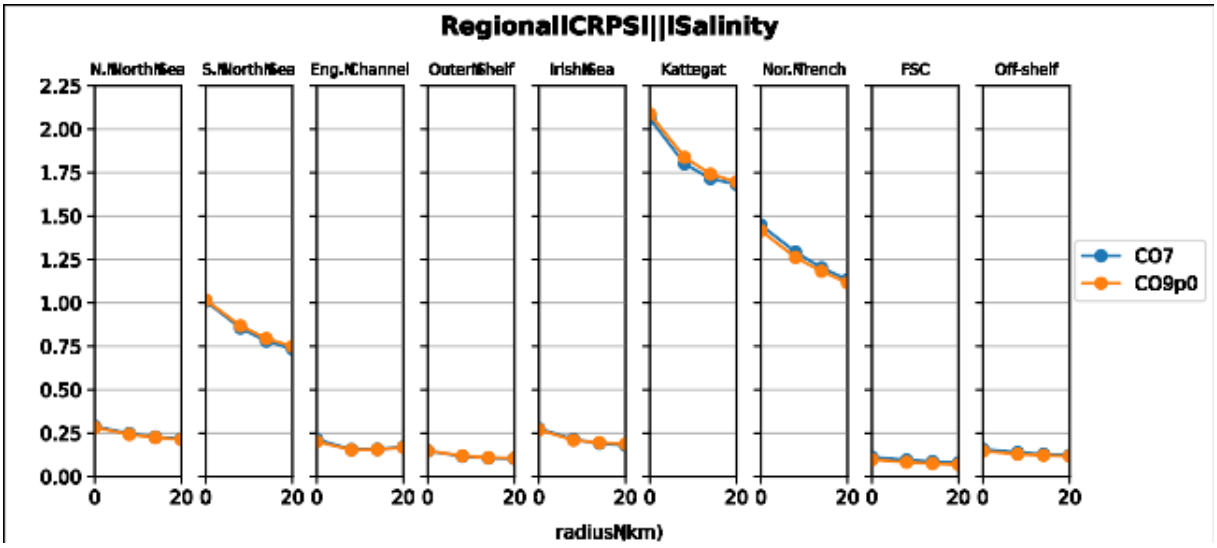

Figure 11. SSS Continuous Ranked Probability Score (CRPS) for eight regions in the AMM domain. Comparisons made between the CO9p0 and CO7 model runs.