OPerational ECology

Ecosystem forecast products to enhance marine GMES applications

DG SPACE

Collaborative Project - small or medium-scale focused research project

Project Number: 283291

Deliverable No: D2.3		Workpackage: 2	
Date:	31.10.2012	Contract delivery due date	31.09.2012
Title:	Target variables and benchmarking metrics: *Technical Specification of Benchmarking Tool v1.0*		
Lead Partner for Deliverable	BC		
Author(s):	Thomas Storm, Norman Fomferra		
Dissemination level (PU=public, RE=restricted, CO=confidential)			PU
Report Status (DR = Draft, FI = FINAL)			DR

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under Grant Agreement No. **283291** for the project **Operational Ecology**: ecosystem forecast products to enhance marine GMES applications (OPEC)

OPEC Overview

"OPEC provides an enhanced capability to predict indicators of good environmental status in European regional Seas"

The OPEC project (Operational Ecology) will help develop and evaluate ecosystem forecast tools to help assess and manage the risks posed by human activities on the marine environment, thus improving the ability to predict the "health" of European marine ecosystems. The programme will focus on four European regional seas (North-East Atlantic, Baltic, Mediterranean and Black Seas) and plans to implement a prototype ecological Marine Forecast System, which will include hydrodynamics, lower and higher trophic levels (plankton to fish) and biological data assimilation.

Products and services generated by OPEC will provide tools and information for environmental managers, policymakers and other related industries, laying the foundations for the next generation of operational ecological products and identification of knowledge / data gaps.

OPEC will use the EU's Global Monitoring for Environment and Security Marine Service as a framework and feed directly into the research and development of innovative global monitoring products or applications. This in turn will advise policies such as the European Marine Strategy Framework Directive and Common Fisheries Policy, as well as the continued monitoring of climate change and assessments of mitigation and adaptation strategies.

www.marineopec.eu

Table of Contents

1. Introduction and Purpose

This Technical Specification frames the system requirements and the design of the benchmarking tool, which is to be implemented in the scope of WP2 of the OPEC project.

Its main purpose is to define input, mode of operation, and output of the benchmarking tool. Additionally, brief sections about runtime environment and user interfaces are included.

As input definition we propose a common model output data format, designed with the goal that all models produce data compliant to that data format. Such a common model output data format is needed in order to enable BC to develop the benchmarking tool. The requirements on that format are listed in section 2.

The mode of operation of the benchmarking tool is based on which benchmark algorithms need to be implemented, and is described in section 3.

The outputs created by the benchmarking tool will be in a certain data format; the specification of this format is to be found in section 4.

Finally, sections 5 and 6 contain information about user interfaces and runtime environment.

2. Reference Documents

Ref	Document Title	Date
[RD-1]	Allen, I.: User guide and report outlining validation methodology (D2.7 of project MEECE)	10/2009
[RD-2]	Saux Picart, S., Butenschön, M., and Shutler, J. D.: Wavelet-based spatial comparison technique for analysing and evaluating two-dimensional geophysical model fields, Geosci. Model Dev., 5, 223-230, doi:10.5194/gmd-5-223-2012, 2012	02/2012

3. Model Output Data Format

This section breaks down into two parts:

In the first part, the model output description documents (MODDs) are assessed with respect to the output format each of them proposes. These different data formats are compared to each other, and similarities as well as differences are identified.

The second part uses the information gathered in the first part to create the common data format BC needs for creating the benchmarking tool.

Assessment of MODDs

This section mainly features Table 1, which is listing the current output formats of the different models. Based on this table the common data format is defined in the next section.

Model vendor	**File format**	**Filename pattern components**	**Directory structure pattern**	**Time Range**
DMI	NetCDF-3	• Model name (*ERGOM*) • region • start date • end date • model version • format version • id	modelout/assim/<region>/<year>/<month>/<day>	2001 - 2010, daily
DTU Aqua	ASCII	*unspecified*	*Unspecified*	1963 - present, yearly, quarterly
HCMR	NetCDF-3	• Model name (*ERSEM* or *PELFISH*) • region • start date • model version • format version • id	modelout/assim/<region>/<year>/<month>/<day>	2004 - 2010, daily
IMS-METU	NetCDF-3	• Model name (*Bims*) • region • start date • end date • model version • format version • id	modelout/assim/<region>/<year>/<month>/<day>	1990 - 2010, daily

OGS	NetCDF-3, CF 1.3	• output date • constant strings(*dm-OGS---opatm_bfm1-MED-b* and *fc-fv02.00*) • run date	Archive/<run-date-lunch[1]>/POSTPROC	2008 - present, daily
PML	NetCDF-3, CF 1.6	• Model name (*ERSEM*) • region • start date • end date • model version • format version • id	modelout/assim/<region>/<year>/<month>/<day>	2004-2010, daily
Summary	As file format, we observe that most models already write NetCDF-3 output. However, we deem it crucial that the models' output is written in a harmonised way. We propose CF-compliant output, since it standardizes the access to precisely the data the benchmarking tool will read.	The filename patterns are all very similar; there should be no problem harmonising them. See Section 3 for a proposed common filename pattern based on the patterns listed in the table.	The same holds for the directory structure patterns.	It is important to note that the earliest outputs are starting in 1963, and there are three different kinds of output: daily, quarterly, and yearly. These observations have influence on the proposed model output data format.

Table 1: MODDs compilation

[1] Probable mistake kept here since we do not know what was intended.

Proposition of an OPEC-Common Data Format

Based on the table provided in Section 2, and following considerations of what is needed for a benchmarking tool that is able to deal with the output of all the different models, we develop the frame for a common data model.

General considerations

1. The file format shall be NetCDF-3 (because most of the model output data is also NetCDF 3).
2. The files shall be compliant to the CF conventions v1.4 or higher.
3. Multiple variables in a single file are allowed only if they are on exactly the same grid and coordinates: we want to have a unique coordinate system in each file.
4. Variables shall be arranged in a regular gridded format in EPSG 4326 projection (http://spatialreference.org/ref/epsg/4326/).
5. If applicable, use scaled integer variables.
6. It is important that a common unit for the time dimension is used. We suggest to use *seconds since 1960-01-01T00:00:00Z UTC* for two main reasons. First, we cannot use the more common year 1970 because some model data will lie before that date. Second, we want to be able to support second-precise time information.
7. The coordinate variables *lat*, *lon*, and *time* shall be of data types *float32*, *float32*, and *int32* respectively.
8. If a variable's dimensions have the interpretations of *date or time (T)*, *height or depth (Z)*, *latitude (Y)*, or *longitude (X)*, we want these dimensions to appear in the relative order *T*, then *Z*, then *Y*, then *X* in the NetCDF header (cp. http://cf-pcmdi.llnl.gov/documents/cf-conventions/1.6/cf-conventions.html, section 2.4)

Reference data

The reference data (typically the in-situ data) needs to be incorporated into the model output files (as agreed upon the OPEC KO in Kopenhagen, Jan 2012).

It shall be incorporated in the product according to the rules about Point Data that have been introduced in CF conventions 1.6 (see appendix H.1, "Point Data"[2]). Most notable of these rules are:

1. There is only one dimension for point data; coordinates are represented by dedicated variables that all share this dimension.
2. The actual geophysical variables need to feature the attribute *coordinates* in order to create a mapping to the coordinate variables.
3. This is a fundamental difference to how model data is represented: while model data is represented in an *n*-dimensional grid, the point data is not gridded at all, but the location in space (which is, for example, given by *lat*/*lon*/*depth*) and time for each reference measurement is given by dedicated variables.

[2] http://cf-pcmdi.llnl.gov/documents/cf-conventions/1.6/cf-conventions.html#idp8294224

Naming conventions

1. The common filename pattern shall include every piece of information that the filename patterns listed in Table 1. Thus, we propose the following harmonised pattern:

```
<modelCode><modelVersion>_<refDataCode>_<vendorCode>{_<regi
onCode>}_<startDate>{-<endDate>}_<id>.nc
```

With:

Filename component	Description	Optional
<modelCode>	An arbitrary model identifier	no
<modelVersion>	An arbitrary string describing the version, such as *1.4a*	no
<refDataCode>	An arbitrary string identifying the reference data source	no
<vendorCode>	An arbitrary identifier for the model developers, such as *OGS*	no
<regionCode>	An arbitrary string denoting the region	yes
<startDate>	The start date in the format *yyyyMMddTHHmmssZ*	no
<endDate>	The end date in the format *yyyyMMddTHHmmssZ*	yes
<id>	An arbitrary id string, may contain both letters and digits	no

This pattern comprises all components most model outputs currently create, except for the OGS model. However, the constant strings that appear in the OGS model output may safely be put into the `<vendor code>` and/or `<model code>` components.

Example filenames:

- `MyChlModel1.1_ICES_BC_20101201000000-20101231235959_0001.nc`
- `MySSTModel2.0_ICES_BC_northsea_20101201000000_someid.nc`

2. The directory structure is not fully homogeneous between the different models, however a harmonised directory structure would significantly simplify the benchmarking tool development. Since the different models write output belonging to different time ranges, we need a rather flexible directory structure pattern. We therefore propose to use a directory structure pattern that allows one to not specify month, day, and region code, since these parameters are not always applicable.

```
modelout/assim{/<region_code>}/<year>{/<month>}{/<day>}/<model_code>/
```

Example directory structures:

```
modelout/assim/2010/MyAnnualSSTModel/<files>
modelout/assim/northsea/2010/MyAnnualSSTModel/<files>
modelout/assim/northsea/2010/01/MyMonthlySSTModel/<files>
modelout/assim/northsea/2010/01/31/MyDailySSTModel/<files>
```

3. The data needs to be distributed within a coordinate system given by at least the dimensions `lat`, `lon`, and `time`, while the coordinates are determined by the identically named coordinate variables. It is crucial that the CF naming conventions regarding these dimensions and variable names are considered.

Attributes

1. The variables need to feature at least the following attributes:
   - *long_name*
   - *standard_name* (if applicable)
   - *_FillValue* (not *missing_value*)
   - *units*

   Of course, additional attributes may be added, such as *_CoordinateAxes*, however, they are not necessarily interpreted by the benchmarking tool.

2. There are a number of global attributes that the output data files are expected to comprise; actually, these are all the global attributes listed in http://cf-pcmdi.llnl.gov/documents/cf-conventions/1.6/cf-conventions.html, sections 2.6.1 and 2.6.2:

`Conventions`	The conventions being used; thus, one of {`CF-1.4`, `CF-1.5`, `CF-1.6`}
`title`	A succinct description of what is in the dataset.
`institution`	Specifies where the original data was produced.
`source`	The method of production of the original data. If it was model-generated, `source` should name the model and its version, as specifically as could be useful. If it is observational, `source` should characterize it (e.g., "`surface observation`" or "`radiosonde`").
`history`	Provides an audit trail for modifications to the original data. Well-behaved generic netCDF filters will automatically append their name and the parameters with which they were invoked to the global history attribute of an input netCDF file. We recommend that each line begin with a timestamp indicating the date and time of day that the program was executed.
`references`	Published or web-based references that describe the data or methods used to produce it.
`comment`	Miscellaneous information about the data or methods used to produce it.

Example file

See Listing 1 for an exemplary NetCDF header file, featuring both modelled data and reference data. The modelled data is four-dimensional and has the four dimensions `time`, `depth`, `lat`, and `lon`. The example modelled variable is `chl`.

The fifth dimension contained in the header is *record_num*. It is the only dimension used by the reference measurements, and is in turn used only by the reference measurements. In order to represent the location of the reference measurements in space and time, dedicated variables are employed, called *time_ref*, *depth_ref*, *lat_ref*, and *lon_ref*. The reference measurement variable is called *chl_ref*. *chl_ref*, as reference measurement variable, features the attribute *coordinates* which lists the names of its coordinate variables.

```
netcdf example_chl {
dimensions:
   time = 202 ;
   depth = 3 ;
   lat = 180 ;
   lon = 360 ;
   record_num = 563 ;
variables:
   int time(time) ;
      time:units = "seconds since 1960-01-01 00:00:00" ;
      time:long_name = "time" ;
      time:standard_name = "time" ;
      time:axis = "T" ;
      time:_CoordinateAxisType = "Time" ;
   float depth(depth) ;
      depth:units = "m" ;
      depth:long_name = "depth" ;
      depth:standard_name = "depth" ;
      depth:positive = "down" ;
      depth:axis = "Z" ;
      depth:valid_min = 0.f ;
      depth:_CoordinateAxisType = "Height" ;
      depth:_CoordinateZisPositive = "down" ;
   float lat(lat) ;
      lat:units = "degrees_north" ;
      lat:long_name = "latitude" ;
      lat:standard_name = "latitude" ;
      lat:axis = "Y" ;
      lat:valid_min = -90.f ;
      lat:valid_max = 90.f ;
      lat:_CoordinateAxisType = "Lat" ;
   float lon(lon) ;
      lon:units = "degrees_east" ;
      lon:long_name = "longitude" ;
      lon:standard_name = "longitude" ;
      lon:axis = "X" ;
      lon:valid_min = -180.f ;
      lon:valid_max = 180.f ;
      lon:_CoordinateAxisType = "Lon" ;
   float chl(time, depth, lat, lon) ;
      chl:_CoordinateAxes = "time depth lat lon" ;
      chl:_FillValue = -1.f ;
      chl:units = "milligram m-3" ;
      chl:long_name = "Concentration of Chlorophyll in sea water" ;
      chl:standard_name = "concentration_of_chlorophyll_in_sea_water";
      chl:coordinates = "time depth latitude longitude" ;
      chl:source = "my_chlorophyll_model" ;
   int time_ref(record_num) ;
      time_ref:units = "seconds since 1960-01-01 00:00:00" ;
      time_ref:long_name = "reference_time" ;
      time_ref:axis = "T" ;
      time_ref:calendar = "standard" ;
      time_ref:_FillValue = -999 ;
   float depth_ref(record_num) ;
      depth_ref:units = "m" ;
      depth_ref:long_name = "depth of the reference observations" ;
      depth_ref:positive = "down" ;
      depth_ref:axis = "Z" ;
      depth_ref:valid_min = 0.f ;
   float lat_ref(record_num) ;
      lat_ref:units = "degrees_north" ;
      lat_ref:long_name = "latitude of the reference observations" ;
      lat_ref:axis = "Y" ;
      lat_ref:valid_min = -90.f ;
      lat_ref:valid_max = 90.f ;
   float lon_ref(record_num) ;
      lon_ref:units = "degrees_east" ;
      lon_ref:long_name = "longitude of the reference observations" ;
      lon_ref:axis = "X" ;
      lon_ref:valid_min = -180.f ;
      lon_ref:valid_max = 180.f ;
   float chl_ref(record_num) ;
      chl_ref:_FillValue = -1.f ;
      chl_ref:units = "milligram m-3" ;
      chl_ref:long_name = "In-situ conc of chlorophyll in sea water" ;
      chl_ref:coordinates = "time_ref depth_ref lat_ref lon_ref" ;
      chl_ref:source = "my_chlorophyll_buoy" ;

// global attributes:
      :Conventions = "CF-1.6" ;
      :title = "some title" ;
      :institution = "institution code" ;
      :references = "links to references" ;
      :source = "method of production" ;
      :history = "audit trail" ;
      :comment = "comment" ;
      :featureType = "point" ;
}
```

Listing 1: Example NetCDF header

4. Algorithms

This section's aim is to describe the algorithms used for benchmarking the model output data against corresponding reference data. These algorithms were agreed upon on the OpEc WP2 Workshop held in Athens at 24-25 April, 2012; they are described in Table 2. Most of the descriptions are more or less directly copied from [RD-1] and [RD-2].
Note that in the formula descriptions, *D* denotes the data, *M* the corresponding model estimate, $\overline{D}$ the mean of the data set, $\overline{M}$ the mean of the model estimates, *N* the total number of model data matches and *n* the *n*th comparison.

Algorithm name	**Description**	**Formula or reference**	**Priority (E,D,O)**[3]
Nash Sutcliffe Model Efficiency	Measure of the ration of the model error to the variability of the data	$ME = 1 - \frac{\sum_{n=1}^{N}(D_n - M_n)^2}{\sum_{n=1}^{N}(D_n - \overline{D}_n)^2}$	**D**
Percentage Model Bias	Sum of the model error normalised by the data	$Pbias = \frac{\sum_{n=1}^{N}(D_n - M_n)}{\sum_{n=1}^{N} D_n} * 100$	**E**
Pearson Correlation Coefficient	Quality index of least quare fitting between model and data	$R = \frac{\sum_{n=1}^{N}(D_n - \overline{D}_n) * (M_n - \overline{M}_n)}{\sqrt{\sum_{n=1}^{N}(D_n - \overline{D}_n)^2 * \sum_{n=1}^{N}(M_n - \overline{M}_n)^2}}$	**E**
RMSE	Measure of goodness of fit between two data sets	$RMSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(D_n - M_n)^2}$	**E**

[3] E = Essential, D = Desirable, O = Optional

unbiased RMSE		$uRMSE\_ = \sqrt{\frac{1}{N}\sum_{n=1}^{N}((M_n)-(\bar{M}_n))(D_n-\bar{D}_n)^2}$	**E**
Reliability Index	Average factor by which model predictions differ from observations	$RI = exp\sqrt{\frac{1}{N}\sum_{n=1}^{N}(log\frac{D_n}{M_n})^2}$	**E**
Taylor Diagrams	Graphically summarise the statistical relationships between two datasets in terms of correlation, centred root-mean-square difference and amplitude of standard deviations.	see Figure 1 for example graphic	**D**
Target Diagrams	Graphically provides information about relationship between two datasets in terms of bias, unbiased and total RMSE	see Figure 2 for example graphic	**D**
Wavelet analysis	Visualises the degree of agreement between spatial features in two datasets	Algorithm described in [RD-2]	**O**
Spatial principal component analysis	Employs PCA, one of the oldest and most widely applied statistical methods, with spatial data	Algorithm TBD	**O**

Table 2: Algorithm overview

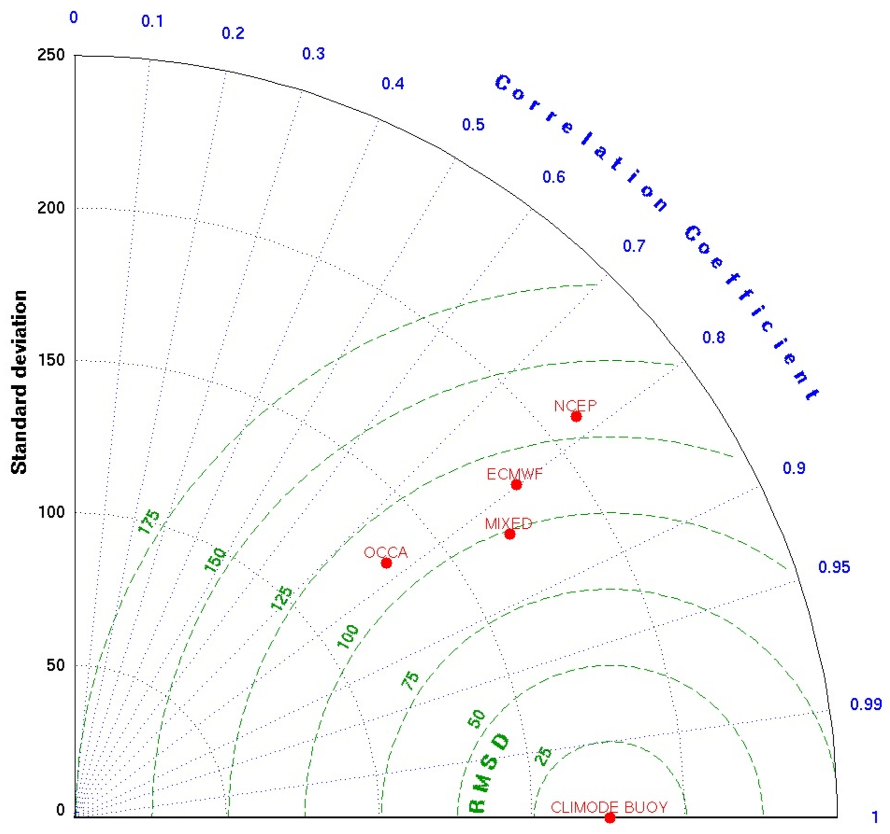

Figure 1: Example Taylor diagram

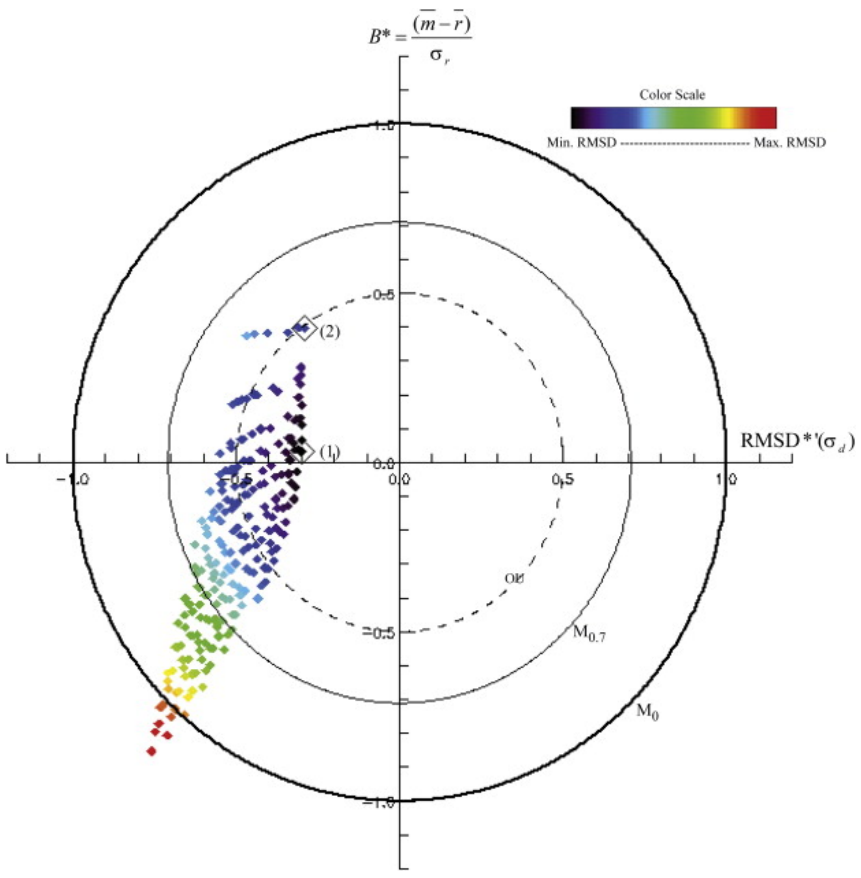

Figure 2: Example Target diagram

5. Benchmarking Output Format

The precise benchmarking output format is yet to be defined, because it directly depends on the algorithms listed in section 3. However, the basic output can already be specified:

- the benchmarking tool will create images (png) of target and taylor diagrams, as well as the ASCII data these diagrams are based on.
- if applicable, the benchmark results will be written as NetCDF files; the format is yet to be defined.
- probably it is possible as well as reasonable that benchmark results are written to CSV files
- if applicable, the benchmark results are written as XML files.

6. User Interface

There are foreseen two different modes of how to use the benchmarking tool: a) the 'standard' mode, which means providing configuration options, running the tool and afterwards inspecting the results, and b) the 'interactive' mode which allows users to dynamically manipulate data and run different benchmarking algorithms from a dedicated Python console.

Note: As discussed during the OPEC KO (Jan 2012), it would be desirable to also implement a web service interface (for machine to machine communication) in addition to the command line interface; however, the implementation of such a service is out of the scope of this project.

The standard mode

The standard interface to the benchmarking tool will be a command-line interface. See Listing 2 for usage of the tool; Listing 3 depicts an exemplary call of the tool.

```
Usage:
opec_benchmark
      [-a <path>]  path to plain text file containing sectioned key/value pairs
                   that specifies algorithms and respective parameters to be used
                   (see Listing 4 for an example); if unspecified, all algorithms
                   are used with default parameters
      [-o <path>]  path to target directory; if unspecified, working directory is
                   used
      [-p <prefix] target name prefix; if unspecified, default prefix is used
      < path>      path to model output file
```

Listing 2: Usage of the benchmarking tool

```
>>> opec_benchmark
      -a    /benchmarking/in/algorithm_config.ini
      -o    /benchmarking/out/
      -p    sst_matchup_2010/01/31
      /modelout/assim/2010/01/31/MyDailySSTModel/northsea/MyChlModel[..].nc
```

Listing 3: Example call of benchmarking tool

```
# comment
[someBenchmarkingAlgorithm]
weight=3.0
threshold=0.5

[someOtherBenchmarkingAlgorithm]
weight=2.3
```

Listing 4: Exemplary file 'algorithm_config.ini'

The interactive mode

The interactive mode means that the tool can be run interactively from a dedicated Python console, e.g. the IPython shell[4]. See Listing 5 for a tentative example of how to use the tool in the shell. Note that this shall generally illustrate the idea of how way to work interactively with the tool, and the actual denominators will probably change.

Legend: Lines in black are python code and comments; lines in dark blue are console output.

[4] http://ipython.org/

```
>>> # import benchmarking tool functionality
>>> import * from opec_benchmark
>>> # create in-memory representation of model file
>>> data = DataStorage("/data/northsea/MyChlModel.nc")
>>> # list model variables
>>> data.list_model_vars()
['chl', ...]
>>> # list reference variables
>>> data.list_ref_vars()
['chl_ref', ...]
>>> # get the data of the geophysical chlorophyll variable
>>> chl = data.vars['chl']
>>> # print it to the console
>>> chl
[3 4 12 2 3 -- 0 2 3 --]
>>> # mask the third value out (because it is considered an outlier)
>>> chl.mask[2] = True
>>> # print the changed data to the console
>>> chl
[3 4 -- 2 3 -- 0 2 3 --]
>>> # compute general statistics based on the read data
>>> statistics = compute_basic_statistics(data)
>>> # print rmse to console
>>> statistics.rmse
0.3
>>> # compute taylor diagram and receive result both as image and array
>>> taylorDiagramImage, taylorDiagramData = compute_taylor_diagram(data)
>>> # print taylor diagram array
>>> taylorDiagramData
[0.2   0.3   0.15]
>>> # write taylor diagram to disk
>>> write_taylor_diagram(taylorDiagramData, "/data/northsea/taylor.png")
```

Listing 5: Interactive Benchmarking

7. Runtime Environment

As decided unanimously during the OPEC KO meeting, the benchmarking tool will be written in the programming language *Python*[5], which is widely used for scientific purposes. The APIs for scientific

[5] http://www.python.org/

computing, *NUMPY* and *SCIPY*[6], are also used within the benchmarking tool. The NetCDF IO is handled using the *netcdf4-python*[7] API. In order to create the output graphics, the library *matplotlib*[8] is being considered. The project will feature many unit-level tests; the framework in which these tests are performed is the *unittest* [9]framework with the *nose* extension. As intermediate storage facility *PyTables* [10]is employed.

The implementation in Python guarantees platform independence, so the benchmarking tool can be run on any OS.

Depending on the amount of data that is to be benchmarked, the tool may also run on Calvalus[11], using the Apache Hadoop[12] implementation of Google's Map/Reduce[13] algorithm.

8. Deliverable completion

In Annex 1 Description of Work the Benchmarking Tool was intended to be delivered by October 2012 (M9). More time than originally anticipated was necessary to broadly gather user requirements, and particularly the need of a dedicated Technical Specification document was not foreseen, as a result the Benchmarking Tool could not be submit in M9. Rather, the Technical Specification has been submitted as first part of D2.3. The full Deliverable will be complete by M15.

[6] http://numpy.scipy.org
[7] http://code.google.com/p/netcdf4-python/
[8] http://matplotlib.sourceforge.net/
[9] http://docs.python.org/2/library/unittest.html
[10] http://www.pytables.org
[11] http://www.brockmann-consult.de/calvalus
[12] http://hadoop.apache.org/
[13] http://research.google.com/archive/mapreduce.html