Dear Andrew,
Dear reviewers,

please find our detailed answers to the referees' letters below.

General remarks:

1. In this text, we used boldface for our answers and italics for the reviewers' original comments.
2. Line numbers refer to the included revised document.
3. As suggested by the second reviewer, we have used professional language editing support for this revision. Thus, the provided latex-diff document includes also many changes resulting from this process.
4. Figure/Table numbering: We experienced problems with figure and table numbering: After adding an appendix, we received the latex compilation error

   ```
   ! LaTeX Error: Too many unprocessed floats.
   ```

   Using the\clearpage command, the figure numbering is wrong (starting from Fig. 21 etc.). Alternative would be not to use the „\appendix" command. This has to be sorted out with the copernicus publishers.

Referee #1:

This is a review of the second version of the manuscript : "Metos3D: A Marine Ecosystem Toolkit for Optimization and Simulation in 3-D – Simulation Package v0.3.2" by Piwonski and Slawig.

The manuscript has certainly improved from the previous version, the objectives are now outlined and it clearly transpires the effort made by the authors to improve its structure and readability. However, in its present form this manuscript is not ready yet for publication in GMD until further moderate changes are made to the text to eliminate most of the residual confusion. As a general comment, I think the lack of coherence in the terminology used throughout the paper is the reason of the "fugacity" of the main message that I have perceived. In other words, I think that if the authors try to call things the same way throughout the text, after having clearly defined them (if possible) things would improve significantly.

Finally, I would like to call the attention of the authors to something that would have made this reviewer's task much easier at little cost. The font chosen for the document where the answers to the reviewer's comments are reported is an incredibly poor one. This, together with the plain editing of the text made reading such document nothing short of painful. Latex is great but sometimes a more popular text editor can do wonders when it comes to

highlighting text, using bold font etc etc; all things usually appreciated for these types of documents. Below are some comments and suggestions:

Main points:

-Section 1, page 1, lines 64-66. Here three strategies used to accelerate the computation of steady-state are mentioned as they are put together in Metos3D. I was expecting later in the text to find somehow a tighter correspondence to this outline in the organization of the sections/subsections but the correspondence wasn't always obvious to me. Again, I think it could all be explained in a much more linear way when the terminology is well-defined in the introduction and it is used in a coherent way throughout the text.
We think that the ordering of the Sections 3. Off-line simulation, 4. Steady annual cycles (containing Newton method) and 5. Software description (containing spatial parallelization) corresponds now better to the outline in the introduction.

For example: it wasn't obvious what you were comparing Metos3D with in Section 7.4. Here you use the expression : "parallel performance of the TMM" but you never explain what you exactly mean by it.
Here scalability is meant, which is now mentioned explicitly, see line 749.

I am familiar with the TMM and it took a while to me to understand that you were comparing Metos3D with the implementation provided by Khatiwala together with the transport matrices you use here. In the Introduction you briefly describe this comparison at lines 133-136 where you use the expression :" the one used in Khatiwala (2013)". Here is where you should assign to "the one used in Khatiwala (2013)" a name and stick to it in the rest of the manuscript.
The name „TMM framework" has been introduced for this purpose now, see line 82.

-Section 6.1. It is an improvement from the previous version. The use of the schematic in Figure 1 (note that numbering of the figures start from 11) ...
Figure numbering: see above.

... helps to follow the description of the implementation however, the terminology used in this section does not correspond with that used in the figure. For example, at lines 470 and 472, the words "debug" and "utilization" are used in italics and are actually called "layers" however there's no trace of them in the Figure. In general, as a suggestion , I would simplify the description, try to outline the main message of this section and leave the details for the appendix.
Section 5.1 has been renamed and rewritten. We replaced the old schematic figure by two new ones showing the software layers (Figure 21) and the call graph (Figure 22). See lines 420 ff.

-Section 6.3 concerns only the interpolation of the transport matrices so it should be specified

in the title. Alternatively, you could group all the following parts concerning interpolation under this section. For example Section 6.4 lines 555-569. Also Section 6.5 seems like it could be merged (and shortened with Section 6.3.
As suggested, we merged all text passages regarding interpolation into one ‚Interpolation' section. See lines 550 ff.

-Section 7.3. This part of the analysis is very interesting and very useful for model developers however, I believe it would be useful to present results also in terms of the incremental computing time per tracer vs number of tracers.
We added a new table and the following text passage: (lines 736-738)
„Additionally, in Table 210 the absolute timings and the computing time per tracer versus number of tracers are shown."

Minor comments:

-Introduction, page 1, lines 45-50. These two sentences are incomprehensible to me. I don't understand what is "its intended (intented in the text) later usage", perhaps try to be more explicit. What does it mean "and mentioned in the name of" ? It seems like this bit of text got lost in there somehow. Rephrase all this part with a clear structure.
Has been reformulated. See lines 47-54.

-Introduction, page 2, line 105. "Except for the latter....." what latter?
We reformulated this passage and made clear it refers to a load-balancing algorithm. See line 110.

-Section 5, line 430. Maybe you mean "current" instead of "actual"?
We changed it to ‚common', see line 527.

-Section 6.3, lines 528-535. How is this different from what is commonly done (I guess in the Khatiwala implementation) ? Maybe try to explain (explicitly) how this procedure is different from the common practice and why is preferred.
In general, it is the same what is done in the TMM framework. We just thought it is worth mentioning here as in the TMM references it is not. We used the following references:
[Khatiwala et al., 2005]
Accelerated simulation of passive tracers in ocean circulation models
[Khatiwala, 2007]
A computational framework for simulation of biogeochemical tracers in the ocean
[Khatiwala, 2008]
Fast spin up of Ocean biogeochemical models using matrix-free Newton-Krylov

-Be careful with the order of the Figures as they are mentioned in the text, for example, at line 728 you mention Figure 19 before Figures 14 1nd 15.
The figures are in the right order now.

-Section 6.4, lines 570-579. Why are there two different "data alignments" ? I could not figure out what you mean here. Maybe this should have resulted clear from the previous sections but it did not so this paragraph sounds like coming out of nowhere to me. Also in this paragraph you mention the "software utilization layer" of which there is no trace in Figure 1.

We added a new Section 5.2 ‚Geometry information and data alignment ', which makes this clearer now. See lines 448 ff, in particular lines 462-473:

> We denote by $\boldsymbol{y}_{i,k} \in \mathbb{R}^{n_{x,k}}$ the values of the i-th tracer corresponding to the k-th profile at fixed time step. Then the vector of *all* tracers at a fixed time, here denoted by $\boldsymbol{y}$ omitting the time index, can be represented in two ways: Either by *first* collecting all profiles for each tracer and *then* concatenating all tracers, namely
>
> $$\boldsymbol{y} = \left[(\boldsymbol{y}_{1,k})_{k=1}^{n_p} \ \cdots \ (\boldsymbol{y}_{n,k})_{k=1}^{n_p}\right], \tag{9}$$
>
> or vice versa, i.e.
>
> $$\boldsymbol{y} = ((\boldsymbol{y}_{i,k})_{i=1}^{n_y})_{k=1}^{n_p}. \tag{10}$$
>
> In order to multiply matrices with tracer vectors, the first variant is preferable. In order to evaluate a water-column based biogeochemical model, the second one is appropriate.

-Section 7.1.1, lines 653-668. This part looks like it could go in the Appendix or directly in the instructions.

Yes. It is part of the Appendix now, see lines 967 ff.

-Section 7.2, lines 738-744. Mind to elaborate a little bit further on the cause of those peaks?

We did, see lines 673 ff.

-Section 7.4, lines 841-842. You should explain clearly here what you mean by theoretical efficiency.

We added a text passage that explains the terms ‚ideal' and ‚theoretical', see lines 777-784:

> Figure 219 depicts ideal, theoretical and actual data for speed-up and efficiency. Here, the term 'ideal' refers to a perfectly parallelizable program and a perfect hardware with no delay on memory access or communication. Regarding the load distribution implemented by us a good (theoretical) performance can be observed over the whole range of processes. This refers again to a perfect hardware except that we distribute a collection of profiles of different length here.

-Lines 978-980. This sentence is not clear. Consider rewording.

The whole passage has been rephrased, see line 917 ff.

Referee #2:

The authors have without doubt clarified and improved the general focus of the paper, I welcome the omission of the somewhat pre-mature optimisation section and in particular the analysis involving a suite of biogeochemical models is a nice addition.

However, while I think the contents are generally adequate, the manuscript is still lacking significantly in terms of clarity and precision. I have the feeling that this is partly due to short- comings in English language and grammar, which may be sorted by language editing support,but it is also due to a somewhat careless effort in elaborating and revisiting the text, which at this point of the submission process is a little concerning, so in its current form I cannot recommend the work for publication in Geoscientific Model Development. I can only re-iterate my final comments in this respect in the previous review step.

In the following I give some examples of my concerns (all line numbers refer to the manuscript version with track changes in the authors response):

I believe the title of the work is inadequate: even if optimisation is the ultimate goal, the work does not currently include it , so the title is misleading.
We have designed and implemented a software system that is able to simulate and optimize marine ecosystem models coupled to ocean transport. We assigned the name Metos3D to this system. The name was chosen to reflect its final purpose.

As mentioned in lines 47-54, a prerequisite for optimization is simulation. Since the description of the simulation package that is the topic of this work obviously already fills a whole paper, we decided to present the optimization package separately. Thus we think that the title of the paper exactly reflects this situation. Moreover, we thought that you (the reviewer(s)) somehow recommended such separation after the first submission.

However, we followed the editor's recommendation and changed the title to: „Metos3D: A Marine Ecosystem Toolkit for Optimization and Simulation in 3-D - Part 1: Simulation Package v0.3.2 -", which hopefully expresses (slightly) more clearly this rationale.

Throughout the main body of the text it appears that all states where treated equally in the analysis, while from some figures and the model descriptions in the appendix it appears that only or mainly inorganic phosphate was considered. This should be clarified.
We added information on the regarded tracer variable in each figure caption. Otherwise all states are treated equally, which has been made clear in Section 3, see lines 206 f..

Lines 29 following: State explicitly first that the tool has been tested with 6 biogeochemical models.

Done, see line 11.

Line 87: I can see that the effort increases, but why would it get more complex?
Changed to computational complexity, see line 58.

Line 132-135: language
We rephrased the paragraph, see lines 100 ff.
Old:

~~However, realistically, simulation of marine ecosystem models~~ No matter whether fixed-point or Newton iteration is used, the necessary multiply repeated simulation of one model year for the marine ecosystem in 3-D is still subject to high performance computing. ~~A parallel~~ Parallel software that employs transport matrices and targets a multi-core distributed-memory architecture requires appropriate data types and linear algebra operations. Additionally, ~~a Newton solver and a load balancing algorithm are needed~~the special ocean geometry with different numbers of vertical layers in different regions is a challenge for standard load balancing algorithms – and a chance for the development of adapted versions with improved overall simulation performance. Except for the latter, ~~an adequate basis for an implementation is made~~ the basis for our implementation is freely available by the Portable, Extensible Toolkit for Scientific Computation library (PETSc; Balay et al., 1997, 2012b), which in turn is based on the Message Passing Interface standard (MPI; Walker and Dongarra, 1996).

New:

Whether fixed-point or Newton iteration is used, high performance computing will be needed for running multiple simulations over one year of model time of a 3-D marine ecosystem. Parallel software employing transport matrices and targeting a multi-core distributed-memory architecture requires appropriate data types and linear algebra operations. The specific geometry of oceans with their varying numbers of vertical layers poses an additional challenge for standard load-balancing algorithms – but also offers a chance of developing adapted versions that will improve overall simulation performance. Except for these adaptations our implementation is based on the freely available Portable, Extensible Toolkit for Scientific Computation library (PETSc; Balay et al., 1997, 2012b), which in turn is based on the Message Passing Interface standard (MPI; Walker and Dongarra, 1996).

Line 144: versions of what?
Versions of load balancing algorithms. See lines 106 ff.

Line 145: it's not clear to me what the latter refers to
We reformulated this passage and made clear it refers to a load-balancing algorithm. See lines 110 ff.

Line 172: is->are
Corrected.

Section 2 is a brief mathematical description of the pdes of the coupled system, but not a description of marine ecosystem dynamics. Title of the section needs changing.
We changed the section title to „Model equations for marine ecosystems“ see line 156

Line 265: While I accept that the overall application of the Neumann condition is good enough in the context of testing this software package, for a realistic implementation of a steady state solution of the annual cycle of marine biogeocheimstry, I'm not sure how reasonable a general Neumann condition is. I would have thought that atmospheric deposition of nutrients and riverine discharges have a role here.
The corresponding paragraph has been extended to describe how this (and also Dirichlet b.c.) can be handled, see lines 193 ff.

Line 274 Kappa is diffusivity, not diffusion, diffusion is the process described by the full term.
Corrected. See line 209.

Lines 185 following: the 128 appears as a general rule here, while I'd expect it to depend on the number of grid points and the strategy of parallelization, which restricted to horizontal domain decomposition. Also it's anticipating results and shouldn't be placed in the introduction.
We omitted the number of processes. See line 139.

Eq. 4: what is z?
It is an arbitrary vector in R^{n_y n_x}. This is stated at line 293.

Line 404: other
Corrected. See line 299.

Line 406: "are equivalent with": I suspect what is meant is that all norms fulfill that condition? Equivalent is a different thing.
We refer to the mathematical definition of norm equivalence. We changed the sentence. See lines 299 ff.

Line 520: which number?
The number of inner iterations. We rephrased the sentence. See lines 387 ff.

Lines 555-556: Unclear what is meant by this sentence, I'd drop it.
Dropped.

Line 565: nx I suppose?
Yes. Corrected.

Line 744-756: I can't find any if the following represented in the figure it refers too up to Lin 754? E.g. what is the bottom layer, what is it's role within the software package?
Section 5.1 has been renamed and rewritten. We replaced the old schematic figure by two new ones showing the software layers (Figure 21) and the call graph (Figure 22). See lines 420 ff.

Line 784 it's not true that it can't be split, but that would require message passing between processes.
Corrected. See lines 601 ff.

Line 788-790 not clear what's meant by its mid in relation to the vector length and how that is used for balancing then?
We reformulated the text. See lines 612 ff.

Line 849 following: sounds like a lot of memory operations to reorganise the data structure in the memory space. Should be possible to avoid this using pointers.
To our knowledge, this is not possible. If you define a Fortran routine like

```
subroutine sub(nz, n, y)
    integer :: nz, n
    real*8  :: y(nz, n)
    …
end subroutine
```

it is expected that y represents a contiguous piece of memory.

Lines 877-879 I don't think there's much value in as adding the code fragment here, there no added information with respect to the equation.
We are not sure to what this comment refers to. If it is Listing 1, i.e. the Fortran 95 implementation of the interface, we think it is valuable for the reader.

Lines 890 following: The analogy to the treatment of interpolation remains unclear here as that section doesn't mention any of those routines.
The interpolation section has been reorganized. See lines 550 ff.

Lines 919-921: This sounds more like the section would be a kind of step-through user guide, rather than a description of the software package as the rest of the text. In fact the rest of the section give a lot of details to enable reproduction of the results. This is great and very useful for interested readers, so I think it would be good to mention this in the introduction of the section rather than introducing it as a presentation of results. In fact there is no results in this section until pg 12. Might be worth splitting this into two sections to separate out the part with the actual results from the experiment description.
The experimental setup is now part of the Appendix. See lines 962 ff.

Line 925 "original implementation" is a bid misleading here as it may sound as it would be an original part of this work, while it was rather introduced in the paper cited shortly afterwards (Dutkiewwicz 2005). I'd suggest to drop the "original"

Here, we again followed the editor's suggestions and reformulated the text to:

In order to test our interface we couple an N, N-DOP, NP-DOP, NPZ-DOP, NPZD-DOP model hierarchy as well as an implementation of Dutkiewicz et al. (2005)'s original bio-geochemical model. The former has been implemented from scratch for this purpose. The corresponding equations are shown in Appendix B. The latter is the model used for the MIT General Circulation Model (cf. Marshall et al., 1997, MITgcm) biogeochemistry tutorial. We will denote it as the MITgcm-PO4-DOP model.

Line 976 "filled in", does that mean it has been set to land?
If so it would be interesting to state the reasoning of this choice.

Yes. This originates in the data provided by Khatiwala. We added this to the text:

Old:

The surface grid of the used domain has a longitudinal and latitudinal resolution of $2.8125°$, which results in 128×64 grid points (cf. Figure 12). Note that the Arctic has been filled in. The depth is divided into 15 vertical layers that are depicted in Table 17. This geometry translates to a (single) tracer vector length of ~~$n_y = 52749$~~ $n_x = 52749$ and the corresponding $n_p = 4448$ profiles. Moreover, the total volume of the ocean is specified as $V \approx 1.174 \times 10^{18}\,\mathrm{m}^3$, whereas the minimal and maximal volume of a grid box is $V_{\min} \approx 8.357 \times 10^{11}\,\mathrm{m}^3$ and $V_{\max} \approx 6.744 \times 10^{13}\,\mathrm{m}^3$, respectively. The temporal resolution is at $\Delta t = 1/2880$, which is equivalent to an (ocean) time step of 3 hours assuming that a year consists of 360 days.

New:

The surface grid of the domain used has a longitudinal and latitudinal resolution of $2.8125°$, which produces 128×64 grid points (cf. Figure 23). Note that the Arctic has been filled in, i.e. set to land. This originates in the data provided at the TMM webpage (cf. Khatiwala, 2013). The depth is divided into 15 vertical layers as described in Table 26. This geometry translates to a (single) tracer vector length of $n_x = 52749$ and to $n_p = 4448$ corresponding profiles. Temporal resolution is at $\Delta t = 1/2880$, which is equivalent to an (ocean) time step of 3 hours, assuming that one year consists of 360 days.

Lines 980-983: What is the relevance of these volumes?

They are used to compute a weighted norm. We dropped them here and used them to compare the solution of spin-up and Newton (cf. Section 6.1 line 658 and Table 29).

Lines 1038-1040: Looking at the figure, I don't understand what the phrase "We observe that the solutions converge to the same difference in between consecutive iterations." means?
This was reformulated. See line 656.

Table 16: What is the difference between the two columns, i.e. that does the V stand for?
It stands for volume. This has been added to the figure catpion.

Figures 110,111,... what happened to the figure numbering?
See remark at the beginning.

Figures 19 and similar: the states used in the formula of the norm are not normalised as far as I can see, so what are the states and units we are looking at in the norm? Is this just phosphate? Is it all states? If it is all, shouldn't there be different weights between different states?
It is phosphate only and the units are mmol P/m^3. We added this information in each caption.

Line 790: Figures 117-115?
The figures are in the right order now.

Lines 1132-1134: It is unclear to me how the Sievertsen work has impacted the profiling capacity in this work.
This seems to be a misunderstanding. The passage has been rephrased.
Old:

This profiling capability was also used as the software was ported by Siewertsen et al. (cf. 2013) to an NVIDIA graphics processing unit (GPU). The authors investigated the impact of the accelerator's hardware on the simulation of biogeochemical models. The work comprises a detailed discussion on peak performance as well as memory bandwidth and includes a counting of floating point operations.

New:

Siewertsen et al. (cf. 2013) also made use of this profiling capacity when porting the software to an NVIDIA graphics processing unit (GPU). The authors investigated the impact of the accelerator's hardware on the simulation of biogeochemical models. Their work comprises a detailed discussion of peak performance and memory bandwidth and includes a counting of floating point operations.

Lines 1143-1147: Does the TMM use the same boundary and initial conditions and time steps? I suppose so, but it might be worth mentioning it.
Yes, the configuration is the same. We added this information. See lines 748 ff.

Lines 1165-1166: "Here, we use the given output, which is the timing for the whole run. Overall, for the calculation of the speed-up and efficiency results we use the minimum

timings for a specific number of cores." Not clear to me.
Rephrased. See lines 758 ff.

Line 1185 How is the theoretical speed-up computed?
We reformulated the text. See lines 779 ff.

Lines 13002-1312 I don't understand the "On one hand ..., on the other hand..." here, isn't the point simply that the implementation of different biogeochemical models underlines the flexibility and generality of the interface?
Rephrased. See lines 859 ff.

Lines 1444-1447: meaning of "whose" is unclear.
The sentence has been omitted and the paragraph has been rephrased. See lines 911 ff.

Lines 1485: Not sure what is meant by the investment in the simulation itself.
Dropped.

A1.1 and A1.2: The formulation that phytoplankton is treated "implicitly" in these models is misleading, when it is actual a free model input parameter and should be treated as such (particularly with view on optimisation!).
We added a remark here and also to the description of the NP-DOP model. However, we sticked to the used formulation to be consistent with Kriest et al (2010). See lines 1061 ff.

Manuscript prepared for Geosci. Model Dev.
with version 2014/09/16 7.15 Copernicus papers of the LaTeX class copernicus.cls.
Date: 21 July 2016

Metos3D: A Marine Ecosystem Toolkit for Optimization and Simulation in 3-D – Part 1: Simulation Package v0.3.2 –

Jaroslaw Piwonski[1] **and Thomas Slawig**[1]

[1]Institute for Computer Science and Kiel Marine Science – Centre for Interdisciplinary Marine Science, Cluster The Future Ocean, Kiel University, 24098 Kiel, Germany. Email: {jpi, ts}@informatik.uni-kiel.de

Correspondence to: Jaroslaw Piwonski (jpi@informatik.uni-kiel.de)

Abstract. We designed and implemented a modular software framework for the off-line simulation of steady cycles of 3-D marine ecosystem models based on the transport matrix approach. It is intended ~~to be used in~~ for parameter optimization and model assessment experiments. We defined a software interface for the coupling of a general class of water column-based biogeochemical models, with six ~~of them~~ models being part of the package. The framework offers both spin-up/fixed-point iteration and Jacobian-free Newton method for the computation of steady states.

The simulation package has been tested with all six models. The Newton method converged ~~with standard setting~~ for four models ~~, and with a change in one~~ when using standard settings, and for two more complex models after alteration of a solver parameter or the initial guess~~for two more complex ones~~. ~~For all considered models, both~~ Both methods delivered the same steady ~~state~~ states (within a reasonable precision) on convergence for all models employed, with the Newton iteration ~~being in general~~ generally operating 6 times faster. ~~For one exemplary model, we investigated the effect~~ The effects on performance of both the biogeochemical and the Newton solver parameters ~~on the performance~~were investigated for one model. ~~We performed a profiling analysis for all considered models , in which~~ A profiling analysis was performed for all models used in this work, demonstrating that the number of tracers had a dominant impact on ~~the~~ overall performance. We also implemented a geometry-adapted load balancing procedure which showed ~~nearly~~ close to optimal scalability up to a high number of parallel processors.

1 Introduction

In the field of climate research ~~, simulation~~ simulations of marine ecosystem models ~~is~~ are used to investigate the carbon uptake and storage of ~~the~~ earth's oceans. The aim is to identify those processes that ~~are involved with~~ play a role in the global carbon cycle. ~~This requires a coupled simulation~~ For this purpose coupled simulations of ocean circulation and marine biogeochemistry are required. In this context, marine ecosystems are ~~understood~~ treated as extensions of ~~the latter~~ biogeochemical systems (cf. Fasham, 2003; Sarmiento and Gruber, 2006). ~~Consequently, we will use both terms synonymously below~~Both terms are therefore used synonymously in this paper. ~~However, whereas the~~ The equations and variables of ocean dynamics are well ~~known~~understood. However, descriptions of biogeochemical or ecological sinks and sources still ~~entail uncertainties concerning~~ contain uncertainties with regard to the number of components and ~~parameterizations~~ to parameterization (cf. Kriest et al., 2010).

~~A~~ To improve this situation a wide range of marine ecosystem models ~~needs~~ need to be validated, i.e. assessed ~~regarding~~ as to their ability to reproduce real world data. This involves a ~~professional~~ thorough discussion of simulation results and, ~~moreover~~before this, an estimation of optimal model parameters for preferably standardized data sets ~~beforehand~~ (cf. Fennel et al., 2001; Schartau and Oschlies, 2003).

~~Optimization methods usually require~~ As a rule hundreds of model evaluations are required for optimization. ~~As a consequence, an environment for optimization of marine ecosystemsthat is intended by (and mentioned in the nameof)our software Metos3D~~ Therefore any optimization environment for marine ecosystems, which our software framework is intended to supply (as suggested by its name),

first and foremost has to provide a fast and flexible simulation framework~~at first~~. ~~On this pre-requisite for an optimization environment we concentrate in this paper, always keeping in mind its later intented usage~~In this paper we will concentrate on this prerequisite and present the simulation package of Metos3D. ~~As a consequence, we impose a high standard of flexibility w. r.t. interchange of models and solvers.~~ An optimization package will be released subsequently.

~~The computational effort of a~~ For any fully coupled simulation, i.e. ~~a~~ simultaneous and interdependent ~~computation~~ computations of ocean circulation and tracer transport in three spatial dimensions, ~~is very high ,~~ very high computational efforts are needed even at low resolution. ~~Moreover, the complexity increases additionally~~ Computational complexity increases still more if annual cycles are investigated, ~~in which one model evaluation involves a long time~~ since each model evaluation then involves long-time integration (the so-called spin-up) until an equilibrium state is reached under given forcing ~~is reached (cf. Bernsen et al., 2008).~~ (cf. Bernsen et al., 2008).

~~Individual~~ Several strategies have been developed to accelerate ~~the~~ computation of periodic ~~steady-states of~~ steady states in biogeochemical models driven by a 3-D ocean circulation (cf. Bryan, 1984; Danabasoglu et al., 1996; Wang, 2001). ~~In this work we combine~~ We have combined three of them in our software, namely ~~the~~ so-called off-line simulation, ~~the option for the~~ optional use of Newton's method for ~~the computation of~~ computing steady annual cycles (as an alternative to ~~a spin-up~~spin-ups) and spatial parallelization with high scalability.

Off-line simulation ~~offers a~~ affords fundamentally reduced computational ~~cost compared to~~ costs combined with an acceptable loss of accuracy. The principle ~~idea~~ is to precompute transport data for passive tracers. ~~Such an approach has been~~ This approach was adopted by Khatiwala et al. (2005) ~~to introduce~~ when introducing the so-called Transport Matrix Method ~~(TMM; Khatiwala, 2013)~~(TMM). The authors ~~make use of~~ used matrices to store ~~results from~~ the results of a general circulation model~~and to apply them later on to arbitrary~~ , which were then applied to biogeochemical tracer variables. This method proved to be sufficiently accurate to gain first insights into the behavior of biogeochemical models at global basin-scale (cf. Khatiwala, 2007). The software implementation used therein we denote as the *TMM framework* from now on. It is available at Khatiwala (2013).

From the mathematical point of view ~~,~~ a steady annual cycle is a periodic solution of a system of (in this case) nonlinear parabolic partial differential equations. This periodic solution is a fixed-point ~~of~~ in the mapping that integrates the model variables over one year of model time. ~~In this sense,~~ Seen in this light a spin-up is a fixed-point iteration. ~~By a straighforward procedure ,~~ Using an uncomplicated procedure this fixed-point problem can be ~~equivalently transformed~~ transformed equivalently into the problem of finding the root(s) of a nonlinear mapping. ~~For this kind of problem,~~

Newton-type methods (cf. Dennis and Schnabel, 1996, Chapter 6) are ~~well known~~ well-known for their superlinear convergence when applied to problems of this kind. ~~In combination~~ When combined with a Krylov subspace approach ~~,~~ a Jacobian-free scheme can be realized that is based ~~only~~ on evaluations of just one model year (cf. Knoll and Keyes, 2004; Merlis and Khatiwala, 2008; Bernsen et al., 2008).

~~No matter whether~~ Whether fixed-point or Newton iteration is used, ~~the necessary multiply repeated simulation of one model year for the marine ecosystemin 3-D is still subject to high performance computing~~high performance computing will be needed for running multiple simulations over one year of model time of a 3-D marine ecosystem. Parallel software ~~that employs~~ employing transport matrices and ~~targets~~ targeting a multi-core distributed-memory architecture requires appropriate data types and linear algebra operations. ~~Additionally, the special ocean geometry with different~~ The specific geometry of oceans with their varying numbers of vertical layers ~~in different regions is a~~ poses an additional challenge for standard ~~load balancing~~ load-balancing algorithms – ~~and a chance for the development of adapted versions with improved~~ but also offers a chance of developing adapted versions that will improve overall simulation performance. Except for ~~the latter, the basis for~~ these adaptations our implementation is ~~freely available by the~~ based on the freely available Portable, Extensible Toolkit for Scientific Computation library (PETSc; Balay et al., 1997, 2012b), which in turn is based on the Message Passing Interface standard (MPI; Walker and Dongarra, 1996).

The objective of this work is to ~~unite the mentioned~~ combine three performance-enhancing techniques (off-line computation via transport matrices, Newton method, and highly scalable parallelization) ~~in~~ in order to produce a software environment ~~with~~ which offers rigorous modularity and complete open-source accessibility. ~~Here, modularity refers to the separation of~~ Modularity entails separating data pre-processing and simulation ~~and the flexibility of coupling~~ as well as the possibility of implementing any water column-based biogeochemical model with ~~minimized implementation~~ minimal effort. For this purpose ~~, we~~ we have defined a model interface that permits the use of any number of tracers, parameters~~as well as~~ , and boundary and domain data. ~~Its flexibility we show by using both an available~~ To demonstrate its flexibility we employed an existing biogeochemical model (Dutkiewicz et al., 2005), ~~taken from~~ part of the MITgcm ocean model, as well as a suite of more complex ~~ones~~models, which is included in our software package. Our software ~~allows for choosing among~~ offers optional use of spin-up/fixed-point iteration ~~and Newton method, where~~ or Newton method; for the latter ~~tuning options are~~ some tuning options were studied. As

a result ~~,~~the work of Khatiwala (2008) could be extended by numerically showing convergence for all six ~~abovementioned models~~ models mentioned above without applying preconditioning. Moreover, a detailed profiling analysis ~~for the simulation with the~~ of the simulation when using different biogeochemical models ~~shows~~ demonstrated how the number of tracers impacts ~~the~~ overall performance. Finally ~~,~~ an adapted load balancing method is presented. ~~It shows nearly optimal scalability up to 128 processes,~~ It shows scalability that is close to optimal and in this respect ~~superiority over~~ is superior to other approaches, including the ~~one used in Khatiwala (2013)~~TMM framework (Khatiwala, 2013).

~~The paper is organized as follows.~~ This paper is structured as follows: In Sections 2 and 3~~we describe the marine ecosystem dynamics and recapitulate~~ , model equations are described, and the transport matrix approach is recapitulated. In ~~Sections~~Section 4 ~~we summarize the two options for the computation of~~ both options for computing steady cycles/periodic solutions ~~, namely the~~ (fixed-point and Newton iteration~~, where~~) are summarized, and for the latter ~~we also discuss~~ some tuning options to achieve better convergence are discussed. In Sections 5 and 6, ~~we describe~~ design and implementation of our software package ~~, and~~ are described, while Section 7 ~~shows ist~~ offers a number of numerical results to demonstrate its applicability and performance~~in several numerical results~~. ~~In~~ Section 8 ~~we draw conclusionsand in~~ presents our conclusions, and Section 9 ~~describe~~ explains how to obtain the source code. ~~In the Appendix , we summarize the model equations and parameter settings of the model suite we~~ The Appendix contains all model equations as well as the parameter settings used for this work~~and that is available together with the~~ ; these are available at the same location as the simulation software.

2 Model equations for marine ecosystems

3 ~~Marine ecosystem dynamics~~

~~We~~ We will consider the following tracer transport model, which is defined by a system of semilinear parabolic partial differential equations (PDEs) of the form

$$\frac{\partial y_i}{\partial t} = \nabla \cdot (\kappa \nabla y_i) - \nabla \cdot (v\, y_i) + q_i(y,u,b,d),\ i = 1,\dots,n_y, \tag{1}$$

on a time interval $I := [0,T]$ and a spatial domain $\Omega \subset \mathbb{R}^3$ with boundary $\Gamma = \partial\Omega$. ~~Here~~ $y_i : I \times \Omega \to \mathbb{R}$ denotes ~~one~~ a single tracer concentration, and $y = (y_i)_{i=1}^{n_y}$ is the vector of all tracers. Since we are interested in long-time behavior and steady annual cycles, we will assume that the time variable is scaled in years. ~~We omit the additional dependency on the~~ For brevity's sake we have omitted the dependency on time and space coordinates $(t, \boldsymbol{x})$ in ~~the notationfor brevity~~our notation.

The transport of tracers in marine waters is determined by diffusion and advection~~which is~~ , which are reflected in the first two linear terms on the right-hand side of (1). Diffusion mixing coefficient $\kappa : I \times \Omega \to \mathbb{R}$ and advection velocity field $v : I \times \Omega \to \mathbb{R}^3$ may either be regarded as given data~~or~~ , or else have to be simulated ~~together with~~ by an ocean model along with (1). Molecular diffusion of ~~the~~ tracers is regarded as negligible compared to ~~the~~ turbulent mixing diffusion. Thus κ and both transport terms are the same for all y_i.

~~The biogeochemical processes in~~ Biogeochemical processes within the ecosystem are represented by the last term on the right-hand side of (1), i.e.

$$q_i(y,u,b,d) = q_i(y_1,\dots,y_n,u,b,d), \quad i = 1,\dots,n_y.$$

~~Often, the functions~~ The functions represented by q_i ~~are~~ will often be nonlinear and depend on several tracers, ~~which couples~~ thereby coupling the system. We will refer to the set of functions $q = (q_i)_{i=1}^{n_y}$ as "the biogeochemical model". ~~This model typically depends also~~ Typically this model will also depend on parameters. In the software ~~we present~~ presented in this paper these parameters are assumed to be constant w. r. t. space and time, i.e. we have $u = \boldsymbol{u} \in \mathbb{R}^{n_u}$. ~~In~~ For the general setting of (1) this assumption is not necessary. Boundary forcing (e.g. insolation or wind speed, defined on the ocean surface as $\Gamma_s \subset \Gamma$) and domain forcing functions (e.g. salinity or temperature of the ocean water) ~~my also enter~~ may also enter into the biogeochemical model. These are denoted by $b = (b_i)_{i=1}^{n_b}, b_i : I \times \Gamma_s \to \mathbb{R}$ and $d = (d_i)_{i=1}^{n_d}, d_i : I \times \Omega \to \mathbb{R}$, respectively.

~~A reasonable setting are homogeneous~~ For tracer transport models, Neumann conditions for ~~all~~ the tracers y_i on the ~~entire~~ boundary Γ are appropriate. ~~Moreover, a function $y_0(\boldsymbol{x}) = (y_i(0,\boldsymbol{x}))_{i=1}^{n_y}, \boldsymbol{x} \in \Omega$, has~~ They may be either homogeneous (when no tracer fluxes on the boundary are present) or inhomogeneous (to account for flux interactions with atmosphere or sediment, e.g. deposition of nutrients and riverine discharges). In the inhomogeneous case, the necessary data have to be provided ~~to solve an initial-boundary-value problem for .~~ as boundary data in b. In Khatiwala (2007, Sect. 3.5) it is shown how the case of tracers with prescribed surface boundary conditions (i.e. Dirichlet conditions) can be treated using the TMM. Then, an appropriate change of the transport matrices is necessary and an additional boundary vector has to be added in every time step.

3 ~~Transport matrix approach~~Off-line simulation using transport matrices

~~The transport matrix method (Khatiwala et al., 2005) is a method that~~

The Transport Matrix Method (Khatiwala et al., 2005) allows fast simulation of tracer

transport~~assuming that the forcing data diffusion~~, assuming that forcing data diffusivity κ and advection velocity v are given. ~~The~~ This method is based on ~~the~~ a discretized counterpart of (1). We introduce the following notation: Let the domain Ω be discretized by a grid $(\boldsymbol{x}_k)_{k=1}^{n_x} \subset \mathbb{R}^3$ and one year in time by $0 = t_0 < \ldots < t_j < t_j + \Delta t_j =: t_{j+1} < \ldots < t_{n_t} = 1$.
This means that there are n_t time steps per year. ~~At~~ For time instant t_j, ~~we denote by~~

- $\boldsymbol{y}_{ji} = (\boldsymbol{y}_i(t_j, \boldsymbol{x}_k))_{k=1}^{n_x}$ denotes the vector of the values of the i-th tracer at all grid points,
- $\boldsymbol{y}_j = (\boldsymbol{y}_{ji})_{i=1}^{n_y} \in \mathbb{R}^{n_y n_x}$ denotes a vector of the values of *all* tracers at all grid points, appropriately concatenated.

We use analogous notations $\boldsymbol{b}_j, \boldsymbol{d}_j$, and $\boldsymbol{q}_j$ for ~~the~~ boundary and domain data ~~as well as~~ and for the biogeochemical terms ~~in~~ at the j-th time step. ~~For the boundary data only~~ Only corresponding grid points are incorporated ~~.~~ for boundary data.

The transport matrix method approximates the discretized counterpart of (1) by

$$\begin{aligned} \boldsymbol{y}_{j+1} &= \mathbf{L}_{imp,j}(\mathbf{L}_{exp,j}\boldsymbol{y}_j + \Delta t_j \boldsymbol{q}_j(\boldsymbol{y}_j, \boldsymbol{u}, \boldsymbol{b}_j, \boldsymbol{d}_j)) \\ &=: \varphi_j(\boldsymbol{y}_j, \boldsymbol{u}, \boldsymbol{b}_j, \boldsymbol{d}_j), \quad j = 0, \ldots, n_t - 1. \end{aligned} \tag{2}$$

The linear operators $\mathbf{L}_{exp,j}, \mathbf{L}_{imp,j}$ represent ~~the~~ those parts of the transport term in (1) that are discretized explicitly ~~and~~ or implicitly w. r. t. time~~, respectively~~. ~~Consequently, these operators~~ These operators therefore depend on the given transport data κ, v and thus on time. The biogeochemical term is treated explicitly in (2) ~~by~~ using an Euler step.

Since ~~the transport effects each tracer separately~~ transport affects each tracer individually and is identical for all of them, both $\mathbf{L}_{exp,j}, \mathbf{L}_{imp,j}$ are block-diagonal matrices with n_y identical blocks $\mathbf{A}_{exp,j}, \mathbf{A}_{imp,j} \in \mathbb{R}^{n_x \times n_x}$, respectively. ~~In Khatiwala et al. (2005) , it is described~~ Khatiwala et al. (2005) describes how these matrices can be computed by running one step of an ocean model ~~for~~ employing an appropriately chosen set of basis functions for ~~a~~ tracer distribution. ~~As a consequence, the partition~~ The operator splitting scheme used in this ocean model therefore determines the partitioning of the transport operator in (1) into ~~the explicit and implicit matrix depends on the operator splitting scheme used in the ocean model~~an explicit and an implicit matrix. ~~Usually diffusion (or a~~ Diffusion (or some part of it) is ~~discretized implicitly,~~ usually discretized implicitly; in our case ~~vertical diffusion only~~this applies only to vertical diffusion. By this procedure ~~,~~ we obtain a set of matrix pairs $(\mathbf{A}_{exp,j}, \mathbf{A}_{imp,j})_{j=0}^{n_t-1}$ ~~is obtained~~, which usually are sparse. To reduce storing ~~effort and to make the method feasible at all, only a smaller number of (~~efforts and increase feasibility only a small number of averaged matrices are stored; in our case monthly ~~) averaged matrices is stored~~averages were used. ~~From these ,~~ Starting from these matrices, for any time instant t_j an approximation of the matrix pair ~~at a time instant t_j~~ is computed by linear interpolation.

~~The integration of the tracers over a model year thus just consists of~~ Thus integration of tracers over one model year only involves sparse matrix-vector multiplications and evaluations of the biogeochemical model. ~~Specifically,~~ In fact the implicit part of ~~the~~ time integration is now pre-computed and contained in $\mathbf{A}_{impl,j}$, which is the benefit of ~~the~~ this method. The approximation error of this method when compared to direct coupled computation is determined by the interpolation of ~~the~~ transport matrices, the linearization of ~~eventually used~~ possibly nonlinear discretization schemes (e.g. flux limiters), and ~~disregarding the influence of the biogeochemistry back onto the circulation fieldsdetermine the approximation error of the method compared to a direct coupled computation~~by discounting the reverse influence of ocean biogeochemistry onto circulation fields.

4 Steady annual cycles

The purpose of the software presented in this paper is ~~the~~ to allow fast computation of steady annual cycles ~~of the considered~~ for the marine ecosystem model under consideration. A steady annual cycle is defined as a periodic solution of (1) with ~~period length~~ a period length of 1 (year), thus satisfying

$$y(t+1) = y(t), \quad t \in [0, 1[.$$

Obviously, the forcing data functions b, d ~~are required~~ need to be periodic as well.

~~For the application of the~~ To apply the transport matrix method ~~,~~ we assume that a set of matrices for one model year (generated ~~with such~~ using this kind of periodic forcing) is available, and that these ~~are interpolated to~~ have been interpolated to obtain pairs $(\mathbf{A}_{exp,j}, \mathbf{A}_{imp,j})$ for all time steps $j = 0, \ldots, n_t - 1$. In the discrete setting, a periodic solution ~~satisfies~~ will satisfy

$$\boldsymbol{y}_{n_t+j} = \boldsymbol{y}_j \quad j = 0, \ldots, n_t - 1.$$

Assuming that the discrete model is completely deterministic, it ~~suffices to satisfy this equation just for~~ is sufficient if this equation is satisfied for just one j. ~~Here, we compare solutions of the respective~~ In this section we will compare the solutions for the first time instants of two succeeding model years. Defining

$$\boldsymbol{y}^\ell := \boldsymbol{y}_{(\ell-1)n_t} \in \mathbb{R}^{n_y n_x}, \quad \ell = 1, 2, \ldots$$

as the vector of tracer values at the first time instant of model year ℓ, a steady annual cycle satisfies

$$\boldsymbol{y}^{\ell+1} = \phi(\boldsymbol{y}^\ell) = \boldsymbol{y}^\ell \text{ in } \mathbb{R}^{n_y n_x} \text{ for some } \ell \in \mathbb{N}, \tag{3}$$

where $\phi := \varphi_{n_t-1} \circ \cdots \circ \varphi_0$ is the mapping that performs the tracer integration (2) over one year. ~~Here we omitted all other arguments except of~~ All arguments except for $\boldsymbol{y}$ have been omitted in the notation. ~~Thus, a~~ A steady annual cycle therefore is a fixed-point of the nonlinear mapping ϕ.

Since condition (3) will never be satisfied exactly in a simulation, we measure ~~the periodicity~~periodicity, using norms on $\mathbb{R}^{n_y n_x}$ for the residual of (3) . We use the weighted Euclidean norm

$$\|\boldsymbol{z}\|_{2,w} := \left(\sum_{i=1}^{n_y} \sum_{k=1}^{n_x} w_k z_{ik}^2 \right)^{\frac{1}{2}}, w_k > 0, k = 1, \ldots, n_x, \quad (4)$$

~~for $\boldsymbol{z} \in \mathbb{R}^{n_y n_x}$~~ with $\boldsymbol{z} \in \mathbb{R}^{n_y n_x}$ indexed as $\boldsymbol{z} = ((z_{ik})_{k=1}^{n_x})_{i=1}^{n_y}$. This corresponds to our indexing of ~~the~~ tracers, see Section 3. If $w_k = 1$ for all k, we obtain the Euclidean norm denoted by $\|\boldsymbol{z}\|_2$. ~~A norm that stronger corresponds~~ A stronger correspondence to the continuous problem (1) is achieved by using the discretized counterpart of the $\left(L^2(\Omega)\right)^{n_y}$-norm, where w_k is set to the volume V_k of the k-th grid box. ~~This norm we denote by $\|\boldsymbol{z}\|_{2,\Omega}$. Orther settings of the~~ We denote this norm by $\|\boldsymbol{z}\|_{2,V}$. Other settings of weights are possible. All these norms are equivalent ~~with~~ in the mathematical sense, i.e. it holds

$$\min_{1 \le k \le n_x} \sqrt{w_k} \|\boldsymbol{z}\|_2 \le \|\boldsymbol{z}\|_{2,w} \le \max_{1 \le k \le n_x} \sqrt{w_k} \|\boldsymbol{z}\|_2.$$

for all $\boldsymbol{z} \in \mathbb{R}^{n_y n_x}$ and all weight vectors $w = (w_k)_{k=1}^{n_x}$ satisfying the positivity condition in Eq. (4).

4.1 Computation by spin-up (fixed-point iteration)

~~Repeatedly applying~~ Spin-up signifies repeated application of iteration step (3)~~or — ~~, in other words~~ — integrating —~~, integration in time with fixed forcing until convergence is reached~~, is termed spin-up~~. It is well known by~~ Based on Banach's fixed-point theorem (cf. Stoer and Bulirsch, 2002) it is well-known that, assuming ϕ is a contractive mapping satisfying

$$\|\phi(\boldsymbol{y}) - \phi(\boldsymbol{z})\| \le L\|\boldsymbol{y} - \boldsymbol{z}\| \quad \text{for all } \boldsymbol{y}, \boldsymbol{z} \in \mathbb{R}^{n_y n_x}$$

with $L < 1$ in some norm, this iteration will converge to ~~the~~ a unique fixed-point for all initial values $\boldsymbol{y}^0$. This result ~~still holds on weaker assumptions~~ holds for weaker assumptions as well (cf. Ciric, 1974). ~~The~~ This method is quite robust, but ~~on the other hand~~ shows only linear convergence which is especially slow for $L \approx 1$. An estimation of $L = \max_{\boldsymbol{y}} \|\phi'(\boldsymbol{y})\|$ is difficult, since it involves the ~~Jacobians~~ Jacobian $\boldsymbol{q}_j'(\boldsymbol{y}_j)$ of the nonlinear biogeochemical model at the current ~~iterates~~iterate. Typically, thousands of iteration steps (i.e. model years) are needed in order to reach a steady cycle (cf. Bernsen et al., 2008). ~~The~~ Moreover, this method offers only restricted options for convergence tuning, the only straightforward one being ~~the choice of a~~ to choose different time steps Δt_j. ~~To to so, the~~ For this all transport matrices have to be re-scaled accordingly. The ~~natural~~ obvious stopping criterion is ~~the~~ reduction of the difference between two succeeding iterates measured by

$$\varepsilon_\ell := \|\boldsymbol{y}^\ell - \boldsymbol{y}^{\ell-1}\|_{2,w}$$

in some – optionally weighted – norm.

4.2 Computation by inexact Newton method

By defining $F(\boldsymbol{y}) := \boldsymbol{y} - \phi(\boldsymbol{y})$, the fixed-point problem (3) can be equivalently transformed into the problem of finding a root of $F : \mathbb{R}^{n_y n_x} \to \mathbb{R}^{n_y n_x}$. This problem can be solved by Newton's method (cf. Dennis and Schnabel, 1996; Kelley, 2003; Bernsen et al., 2008). We apply a damped (or globalized) version that incorporates a line search (or backtracking) procedure which (under certain assumptions) provides superlinear and locally even quadratic convergence. Starting from an initial guess $\boldsymbol{y}^0$, in ~~every~~ each step the linear system

$$F'(\boldsymbol{y}^m)\boldsymbol{s}^m = -F(\boldsymbol{y}^m) \quad (5)$$

has to be solved, followed by an update $\boldsymbol{y}^{m+1} = \boldsymbol{y}^m + \varrho \boldsymbol{s}^m$. ~~Here~~ $\varrho > 0$ ~~is a~~ here denotes the step-size~~that~~, which is chosen iteratively ~~such~~ in such a way that a sufficient reduction in $\|F(\boldsymbol{y}^m + \rho \boldsymbol{s}^m)\|_2$ is achieved (cf. Dennis and Schnabel, 1996, Section 6.3). Note that regarding the Newton solver the Euclidean norm is used. This is determined by the PETSc implementation.

The Jacobian $F'(\boldsymbol{y}^m)$ of F at ~~the current iterate includes~~ any current iterate contains the derivative of one model year, thus it is not as sparse as the transport matrices themselves. ~~As a consequence, a~~ Therefore a matrix-free version of Newton's method is applied: The linear system (5) ~~itself~~ is solved by an iterative, so-called Krylov subspace method, which only requires the evaluation of matrix-vector products $F'(\boldsymbol{y}^m)\boldsymbol{s}$. Since $F'(\boldsymbol{y}^m)$ cannot be expected to be ~~neither symmetric nor~~ symmetric or definite, we use the generalized minimal residual method (GMRES, Saad and Schultz, 1986). The ~~needed~~ matrix-vector products needed for this can be interpreted as directional derivatives of F at ~~the~~ point $\boldsymbol{y}^m$ in ~~direction~~ the direction of $\boldsymbol{s}$. ~~They can~~ They may be approximated by a forward finite difference:

$$F'(\boldsymbol{y}^m)\boldsymbol{s} \approx \frac{F(\boldsymbol{y}^m + \delta \boldsymbol{s}) - F(\boldsymbol{y}^m)}{\delta}, \quad \delta > 0. \quad (6)$$

The finite difference step-size δ is chosen automatically as a function of $\boldsymbol{y}^m$ and $\boldsymbol{s}$ (cf. Balay et al., 2012a). An alternative ~~here~~ method would be an exact evaluation of the derivative using the forward mode of ~~algorithmic differentiation~~ Algorithmic Differentiation (cf. Griewank and Walther, 2008).

~~The above~~ This approximation of the Jacobian or directional derivative is one reason ~~for this method to be called an~~ to call this method *inexact*~~one~~. The second reason is the

fact that the inner linear solver has to be stopped and ~~thus is also~~ therefore also is not exact. ~~Here we~~ We use a convergence control procedure based on the technique described by Eisenstat and Walker (1996) for this purpose. ~~They stop~~ Stopping occurs when the Newton residual at the current inner iterate $\boldsymbol{s}$ satisfies

$$\|F'(\boldsymbol{y}^m)\boldsymbol{s} + F(\boldsymbol{y}^m)\|_2 \leq \eta_m \|F(\boldsymbol{y}^m)\|_2. \tag{7}$$

The factor η_m is determined ~~as~~ by

$$\eta_m = \gamma \left(\frac{\|F(\boldsymbol{y}^m)\|_2}{\|F(\boldsymbol{y}^{m-1})\|_2} \right)^{\alpha}, \quad m \geq 2, \quad \eta_1 = 0.3. \tag{8}$$

This approach avoids so-called over-solving, i.e. wasting inner steps ~~when the current Newton step was not very successful~~if the current outer Newton residual $F(\boldsymbol{y}^m)$ is still relatively big. The latter ~~is typically the case in the beginning of a Newton iteration~~typically occurs at the beginning of Newton iterations. ~~The parameters~~ Parameters γ and α can be used to ~~influence this behavior~~ avoid over-solving by adjusting inner accuracy depending on outer accuracy in a linear ~~and~~ or nonlinear way, respectively. Moreover, ~~they are~~ both parameters provide a subtle way to tune the solver. In contrast to a fixed-point iteration, Newton's method ~~also~~ even in its damped version may ~~only converge~~ possibly converge only with an appropriately chosen initial guess $\boldsymbol{y}^0$. In a high-dimensional problem ~~as our application~~ such as ours (in $\mathbb{R}^{n_y n_x}$), it is a non-trivial task to find such an initial guess if the ~~method with the standard one (e~~standard one used for the spin-up (i.e. a constant tracer distribution) proves unsuccessful. ~~g. the one used in the literature) is not successful. Thus, if an Newton iteration is slow and the above criterion may consequently lead to~~ In cases where the Newton iteration proceeds slowly and the criterion described above yields only a few inner iterations, it ~~makes sense to increase this~~ may be advisable to increase their number by either decreasing γ or increasing α. ~~We will give examples later on where exactly this strategyenables convergence at all~~Below we will give some examples of how convergence may be made possible using this strategy.

~~Concerning the total effort of~~ In order to estimate the total computational effort needed for the inexact Newton solver and ~~in order~~ to compare its efficiency with the spin-up ~~, we first note~~ method, it must be noted that one evaluation of F basically corresponds to one application of ϕ, i.e. to one model year. ~~Thus, each~~ Each Newton step requires one evaluation of F as the right-hand side ~~in of~~ (5). ~~Within~~ The initial guess for the inner linear solver iteration ~~, the initial guess is always taken as~~ is always set at $\boldsymbol{s} = 0$. Thus ~~,~~ no computation is required for the first step. ~~Each~~ For each following inner iteration ~~require sone additional~~ some evaluation of F is required to compute the second term in the numerator of the right-hand side of (6). ~~Additionally, the~~ The line search may ~~require additional eavluations~~ also require additional evaluations of F. ~~In total~~Taken together, the overall number of inner iterations plus the overall number of evaluations ~~in~~ for the line search determine the number of ~~necessary~~ evaluations of F ~~that can~~ necessary for this method, which may then be compared to the ~~necessary model years in~~ number of model years needed for the spin-up.

5 Software description

6 ~~Biogeochemical model interface~~

~~In this context, our main objective is to specify a general coupling between the transport that is induced by the ocean circulation and the biogeochemical tracer model~~Our software is divided into four repositories, namely `metos3d`, `model`, `data` and `simpack`. The ~~aim is to link any model implementation with any number of tracers, parameters~~ first comprises the installation scripts, the second the biogeochemical model source codes and the third all data preparation scripts as well as ~~boundary and domain data to the driver software~~the data themselves. The ~~coupling must additionally fit into an optimization context, and it must be compatible with Algorithmic Differentiation techniques (cf.Section 7).~~

~~Generally, we assume that a tracer model is implemented for a single water column, synonymously called profile in the following~~last repository contains the simulation package, i.e. the transport driver, which is implemented in C and based upon the PETSc library. While we have often used 1-indexed arrays within this text for convenience, within the source code C arrays are 0-indexed and Fortran arrays are 1-indexed. ~~This means no geometrical information on horizontal vicinity of the vertical profiles is preserved in the interface~~All data files are in PETSc format.

5.1 Implementation structure

The implementation of the simulation package is structured in layers as is shown in Figure 21. ~~Moreover, any client model must be able to take up its states from such profiles.~~The layers are organized hierarchically, i.e. each layer provides routines for the layers above. ~~Models that require a horizontal structure for its internal computation require a redefinition of the interface and a change of the internals of the tool~~The foundation of the implementation is the PETSc library with its data types and the implementation of the Newton-Krylov solver.

~~However, this assumption does not constrain the interface for the future.~~ The *bgc* model layer initializes tracer vectors, parameters and boundary and domain data. ~~In fact, the most important non-local biogeochemical processes happen within a water column (cf. Evans and Garçon, 1997)~~It is responsible for the interpolation of forcing data and the evaluation of the biogeochemical model (cf. Section 5.3). The *transport* layer is responsible for reading in the transport

matrices, interpolating them to the current time step and applying them to the tracer vectors.

~~Consequently, throughout this work, each discrete tracer vector is a collection of profiles~~The main integration routine ϕ (cf. Algorithm 1, 2) is located at the *time stepping* layer. ~~It can be understood as a sparse representation of a land-sea cuboid including only wet grid boxes~~On top resides the *solver* layer, which contains the spin-up implementation and the call to the Newton-Krylov solver.

A call graph for the computation of a steady annual cycle is shown in Figure 22. Note that loops are not explicitly shown therein. Calls to initialization and finalization routines are gathered at the beginning respectively end of a simulation run. The ~~geometry~~ former are responsible for memory allocation and storage of data used at run time. The latter are employed to free memory and delete all vectors and matrices.

The dimensions of the used vectors and matrices depend on the underlying geometry (cf. Section 5.2). The distribution of the work load for a parallel run is determined during initialization of the work load (cf. Section 5.2).

5.2 Geometry information and data alignment

Geometry information is provided as a 2-D land-sea mask ~~with additional~~ plus a designation of the number of vertical layers~~(~~, i.e. the depth of the different water columns (or *profiles*, cf. Figure 23). ~~Hence, a vector length n_y is a sum of non-equidistant~~ This can be understood as a sparse representation of a land-sea cuboid including only wet grid boxes. Hence, the length n_x of a single tracer vector (at fixed time) is the sum of the lengths of all profiles, i.e.

$$n_x = \sum_{k=1}^{n_p} n_{x,k}\,,$$

where n_p is the total number of profiles in the ocean and $(n_{x,k})_{k=1}^{n_p}$ ~~is a~~ the set of profile ~~depths.~~ lengths. Each profile corresponds to a horizontal gridpoint. Due to the locally varying ocean depth, the profile lengths depend on the horizontal coordinate, i.e. on the index k.

~~The~~ We denote by $\boldsymbol{y}_{i,k} \in \mathbb{R}^{n_{x,k}}$ the values of the i-th tracer corresponding to the k-th profile at fixed time step. Then the vector of *all* tracers at a fixed time, here denoted by $\boldsymbol{y}$ omitting the time index, can be represented in two ways: Either by *first* collecting all profiles for each tracer and *then* concatenating all tracers, namely

$$\boldsymbol{y} = \left[(\boldsymbol{y}_{1,k})_{k=1}^{n_p} \dots (\boldsymbol{y}_{n,k})_{k=1}^{n_p}\right], \tag{9}$$

or vice versa, i.e.

$$\boldsymbol{y} = ((\boldsymbol{y}_{i,k})_{i=1}^{n_y})_{k=1}^{n_p}. \tag{10}$$

In order to multiply matrices with tracer vectors, the first variant is preferable. In order to evaluate a water-column based biogeochemical model, the second one is appropriate.

As a result, all tracers need to be copied from representation (9) to (10) after a transport step. After evaluation of the ~~whole n_y tracer model~~for a fixed time index j ~~consist then~~ biogeochemical model we reverse the alignment for the next transport step.

The situation is similar for domain data. Again, we group all domain data profiles by their profile index k, i.e.

$$\left[(\boldsymbol{d}_{1,k})_{k=1}^{n_p} \dots (\boldsymbol{d}_{n_d,k})_{k=1}^{n_p}\right] \longrightarrow ((\boldsymbol{d}_{i,k})_{i=1}^{n_d})_{k=1}^{n_p}$$

where $\boldsymbol{d}_{i,k}$ denotes a single domain data profile. However, no reverse copying is required here.

Boundary data have to be treated in a slightly different way. Here we align boundary values, which are associated with the surface of one water column each,

$$\left[(b_{1,k})_{k=1}^{n_p} \dots (b_{n_b,k})_{k=1}^{n_p}\right] \longrightarrow ((b_{i,k})_{i=1}^{n_b})_{k=1}^{n_p}$$

where $b_{i,k}$ denotes a single boundary data value as opposed to a whole profile. As with domain data, no reverse copying is required.

5.3 Biogeochemical model interface

One of our main objective in this work is to specify a general coupling interface between the transport induced by ocean circulation and the biogeochemical tracer model. We wish to provide a method to couple any biogeochemical model implementation using any number of tracers, parameters and boundary and domain data to the software that computes the ocean transport. Despite the fact that we consider off-line simulation using transport matrices in this paper only, the interface shall not be restricted to this case. This coupling shall furthermore fit into an optimization context, and it shall be compatible with Algorithmic Differentiation techniques (cf. Section 7).

The only restriction we make for the tracer model is that it operates on each single water column (or *profile*) separately. This means that information on exactly one profile is exchanged via the coupling interface. For models that require information on other profiles (e.g. in the horizontal vicinity) for internal computations, a redefinition of the interface and some internal changes would be necessary. In fact, most of the relevant non-local biogeochemical processes take place within a water column (cf. Evans and Garçon, 1997).

The evaluation of a water-column based biogeochemical model for any fixed time t consists of separate model evaluations for each profile ~~. For a fixed~~ (corresponding to a horizontal spatial coordinate), i.e. for profile index k~~we compute~~:

$$\Delta t\,(\boldsymbol{q}_i(t_j, (\boldsymbol{y}_{ii,k})_{i=1}^{n_y}, \boldsymbol{u}, (_ib_{i,k})_{i=1}^{n_b}, (\boldsymbol{d}_{ii,k})_{i=1}^{n_d}))_{i=1}^{n_y}\,. \tag{11}$$

Here, ~~$(\boldsymbol{y}_i)_{i=1}^{n}$~~ $(\boldsymbol{y}_{i,k})_{i=1}^{n_y}$ is an input array of n_y ~~profiles~~tracer profiles according to (10), each with a length or depth of

$n_{x,k}$~~,~~. The vector $\boldsymbol{u}$ ~~a vector of~~ contains n_u parameters~~,~~ ~~$(\boldsymbol{b}_i)_{i=1}^{n_b}$~~. Boundary data $(b_{i,k})_{i=1}^{n_b}$ are given as a vector of n_b ~~boundary data valuesand $(\boldsymbol{d}_i)_{i=1}^{n_d}$ an~~ values, and domain data $(\boldsymbol{d}_{i,k})_{i=1}^{n_d}$ as input array of n_d ~~domain data~~ profiles. ~~Both inputs are regarded as already interpolated. The result~~ is Results of the biogeochemical model are stored in the ~~the output array $(\mathbf{q}_i)_{i=1}^{n_y}$ that consist~~ output array $(\boldsymbol{q}_{i,k})_{i=1}^{n_y}$ which also consists of n_y profiles~~as well. Formally , the~~.

Formally speaking this tracer model is scaled ~~with the (ocean~~from the outside by the (ocean circulation) time step~~from the outside~~. However, we ~~integrate~~ have integrated Δt into the interface as a concession to the ~~actual practice , where~~ common practice of refining the time step ~~is often refined~~ within the tracer model implementation (cf. Kriest et al., 2010). ~~Consequently~~As a consequence, the responsibility ~~to scale the result before returning it back~~ for scaling results before returning them to the transport driver software rests with the model implementer.

Listing 1 shows a realization of the biogeochemical model interface in a Fortran 95 subroutine called `metos3dbgc`. The arguments are grouped by ~~their~~ data type. The list begins with variables of the type `integer`, i.e. n_y, $n_{x,k}$, n_u, n_b and n_d. ~~They~~ These are followed by `real*8` (double precision) arguments, i.e. Δt, $\boldsymbol{q}$, t_j, $\boldsymbol{y}$, $\boldsymbol{u}$, $\boldsymbol{b}$ and $\boldsymbol{d}$. ~~We neglected~~ For clarity we have omitted the profile index k and the time index j in ~~the notationfor clarity~~our notation. Moreover, we ~~use~~ have used `dt` as a textual representation of Δt.

~~Additionally, a~~ A model initialization and finalization interface is also specified. The former is ~~denoted~~ named `metos3dbgcinit` and the latter `metos3dbgcfinal`. These routines are called at the beginning of ~~a~~ each model year, i.e. at t_0, and after the last step of the annual iteration, respectively. Both ~~have~~ routines employ the same argument list as `metos3dbgc`~~and~~. They are not shown here. ~~All three routine names~~ The names of all three routines are arbitrary and can be ~~changed~~ altered using pre-processor variables that are defined within ~~the~~ `Makefile`.

6 ~~Software implementation~~

~~The toolkit is divided into four repositories, namely `metos3d`, `model`, `data` and `simpack`. The first comprises the installation scripts, the second the biogeochemical model source codes and the third all the data preparation scripts as well as the data. The latter repository consist of the simulation package,i.e. the transport driver, which is implemented in C and based upon the PETSc library.~~

~~The simulation context is represented by a data type called `metos3d` that gathers all variables. Regarding the biogeochemical models, C, C++ and Fortran implementations are accepted (cf. Section A1). Overall, whereas we often used 1-indexed arrays within the text for convenience, within the source code C arrays are 0-indexed and Fortran arrays are 1-indexed. Moreover, all data files are in PETSc format.~~

5.1 ~~Layers~~

~~The implementation is structured in layers according to which the source files are named. A schematic is shown in Figure ??. The bottom layer is the *debug* layer which implements output formatting and timing routines. Above resides the *utilization* layer. It provides basic routines for reading in options, allocating memory as well as reading data from and writing data to disc. The option system and the individual options are described in the documentation that is located in a subdirectory of the `git` repository of the simulation package. Moreover, the utilization layer comprises routines to arrange profiles within a vector (cf. Section ??) and to compute interpolation factors and indices (cf. Section 5.1) as well. The 2-D land-sea mask is read in by the *geometry* layer and the profiles are balanced by the work *load* layer (cf. Section 5.2).~~

~~The next two layers are the building blocks of the simulation. The *bgc* model layer initializes tracer vectors, parameters as well as boundary and domain data. It is responsible for the rearrangement of the profiles, the interpolation of the forcing data and the evaluation of the biogeochemical model using the interface (cf. Section ??). The *transport* layer is responsible for reading in the transport matrices, their interpolation to the current time step and their application to the tracer vectors (cf. Section ??).~~

~~The next layer is the *time stepping* layer, where the main integration routine ϕ is located (cf. Algorithm 1). The Newton residual F is implemented here as well. On top resides the *solver* layer, which consist of the spin-up implementation and the call to the Newton-Krylov solver provided by PETSc.~~

~~Additionally, all calls to initialization respectively finalization routines are located at the *init* source file. The former are responsible for memory allocation and storage of data used at run time. The latter are employed to free memory as well as delete the used vectors and matrices.~~

5.1 ~~Load balancing~~

~~Once the geometry information is read in, the profiles have to be distributed among the available processes. However, a tracer vector is a collection of *non* equidistant profiles and the biogeochemical models that we couple to the transport matrices operate on whole water columns. Thus, a profile can not be split when the work load is distributed.~~

~~For this case, no suitable load balancing algorithm is provided by the PETSc library. Here, we use an approach that is inspired by the idea of space filling curves presented by Zumbusch (1999). For every profile, we compute its mid in relation to the vector length and scale this ratio by the number of processes. We round this figure down to an integer and~~

~~use the result as the index of the process the profile belongs to. This information is sufficient to consecutively assign the profiles to the processes later on.~~

~~The calculation for 0-indexed arrays is depicted by Algorithm 3. Its theoretical and actual performance is discussed in Section 6.3 where we show results of speedup tests that we performed on two different hardware architectures.~~

5.1 Interpolation

~~The transport~~ Transport matrices as well as ~~the~~ boundary and domain data vectors are provided as sets of files. ~~Although, most of the data we use in this work represents a monthly mean, the~~ The number of files in each set is arbitrary, although most of the data we use in this work represent a monthly mean.

~~Regarding the transport, we have $(\mathbf{A}_{imp,j})_{j=1}^{n_{imp}}$ and $(\mathbf{A}_{exp,j})_{j=1}^{n_{exp}}$, where n_{imp} and n_{exp} specify the number of implicit and explicit matrix files, respectively. Note, we will not assemble both (block diagonal) system matrices during the simulation to avoid redundant storing. Instead, we use the provided matrices to build only a block for each matrix type. The transport is then applied as a loop over separate tracer vectors as explained in Section ??.~~

~~Concerning the boundary and domain forcing, we denote the data files by $((\boldsymbol{b}_{i,j})_{j=1}^{n_{b,i}})_{i=1}^{n_b}$ and $((\boldsymbol{d}_{i,j})_{j=1}^{n_{d,i}})_{i=1}^{n_d}$. Here, n_b is the number of distinct boundary data sets and $n_{b,i}$ is the number of data files provided for the ith set. Accordingly, n_d denotes the number of domain data sets and $n_{d,i}$ is the number of data files of a particular set.~~

However, ~~the time step count~~ time step counts per model year ~~is~~ are generally much higher than the number of available data files. ~~Thus, the~~ For this reason matrices and vectors are ~~*linearly*~~ linearly interpolated to the current time step during ~~the~~ iteration. ~~The~~ All files of a specific data set are interpreted as averages of the time intervals they represent. ~~Consenquently, we interpolate in between the associated centers of these~~ We therefore interpolate between the centers of associated intervals. The appropriate weights and indices are computed on the fly using Algorithm 4.

With regard to boundary and domain forcing, we denote data files by $((\boldsymbol{b}_{i,j})_{j=1}^{n_{b,i}})_{i=1}^{n_b}$ and $((\boldsymbol{d}_{i,j})_{j=1}^{n_{d,i}})_{i=1}^{n_d}$. ~~Both building blocks of the simulation, i. e. the biogeochemical model and the transport step access the interpolation routine in every time step t_j to form a linear combination of the user provided data~~ Here, n_b is the number of distinct boundary data sets, and $n_{b,i}$ is the number of data files provided for the i-th set.

5.2 ~~Biogeochemical model step~~

~~During a simulation the `BGCStep` routine in Algorithm 2 is responsible for the evaluation of the biogeochemical model. For this, the boundary and the domain data must be interpolated first. Here, for~~ In the same way, n_d denotes the number of domain data sets and $n_{d,i}$ the number of data files of a particular set.

For every index i and ~~the~~ its corresponding boundary data set $(\boldsymbol{b}_{i,j})_{j=1}^{n_{b,i}}$ we compute the appropriate weights α, β as well as indices j_α, j_β and ~~form the linear combination as~~ then form a linear combination

$$\boldsymbol{b}_i = \alpha\, \boldsymbol{b}_{i,j_\alpha} + \beta\, \boldsymbol{b}_{i,j_\beta}.$$

The same applies ~~for the~~ to domain data, i.e. for every domain data set $(\boldsymbol{d}_{i,j})_{j=1}^{n_{d,i}}$ we compute

$$\boldsymbol{d}_i = \alpha\, \boldsymbol{d}_{i,j_\alpha} + \beta\, \boldsymbol{d}_{i,j_\beta}.$$

~~Technically, we use the~~ We use PETSc routines `VecCopy`, `VecScale` and `VecAXPY` for this ~~purpose, which is analogous to the interpolation of the transport matrices in Section ??~~process.

~~Next, we rearrange the forcing data and the tracer vectors. This is necessary since the combination of transport matrices and water column models results in two different data alignments. For the application of a matrix to a tracer vector, all profiles of a tracer are kept one behind the other. In contrast, to evaluate the tracer model the same profile of each tracer must be kept in a contiguous piece of memory. Accordingly, this has an effect on the forcing data as well. The routines for rearrangement are provided within the softwares utilization layer.~~

~~Concerning the tracers, we need to copy from n separate vectors to one (block diagonal) vector, where the profiles are grouped by their index, i.e.~~

$$\left[(\boldsymbol{y}_{1,k})_{k=1}^{n_p} \ \dots\ (\boldsymbol{y}_{n,k})_{k=1}^{n_p}\right] \quad \longleftrightarrow ((\boldsymbol{y}_{i,k})_{i=1}^{n})_{k=1}^{n_p},$$

With regard to transport we have $(\mathbf{A}_{imp,j})_{j=1}^{n_{imp}}$ and $(\mathbf{A}_{exp,j})_{j=1}^{n_{exp}}$ as data files, where ~~$\boldsymbol{y}_{i,k}$ denotes the kth profile of the ith tracer. Moreover, after the evaluation of the biogeochemical model we reverse the alignment for the transport step. The same situation occurs regarding the domain data~~ n_{imp} and n_{exp} specify the number of implicit and explicit matrix files, respectively. ~~Again, we group the domain data profiles by their profile index k, i.e.~~

$$\left[(\boldsymbol{d}_{1,k})_{k=1}^{n_p} \ \dots\ (\boldsymbol{d}_{n_d,k})_{k=1}^{n_p}\right] \quad \longrightarrow ((\boldsymbol{d}_{i,k})_{i=1}^{n_d})_{k=1}^{n_p}$$

~~where $\boldsymbol{d}_{i,k}$ denotes a domain data profile. However, no reverse copying is required here.~~

~~The boundary data is a slightly different case. Here, we align boundary values, at which each is associated with the surface of a water column, i.e.~~

$$\left[(b_{1,k})_{k=1}^{n_p} \ \dots\ (b_{n_b,k})_{k=1}^{n_p}\right] \quad \longrightarrow ((b_{i,k})_{i=1}^{n_b})_{k=1}^{n_p}$$

~~where $b_{i,k}$ denotes a single boundary data value in contrast to a whole profile. Analogously to the domain data, no reverse copying is required in this case.~~

Subsequent, we loop over all profiles and evaluate the biogeochemical model for every water column formally using the interface introduced in . Within the implementation, since we only couple models that are written in Fortran, we use the programming counterpart depicted in Listing 1. Finally, as already mentioned, we prepare the output for the transport step.

5.2 Transport step

The application of the transport matrices to tracer variables is the second building block of the simulation. The individual steps are combined in the `TransportStep` routine, which is applicable to both matrix types as shown in Algorithm 2. On entry, we interpolate the user provided Analogous to the interpolation of vectors we first interpolate all user-provided matrices to the current point in time t_j first, i.e. we assemble

$$\mathbf{A} = \alpha \mathbf{A}_{j_\alpha} + \beta \mathbf{A}_{j_\beta}$$

with using the appropriate α, β and j_α, j_β. Analogously to the interpolation of vectors we We use the matrix variants `MatCopy`, `MatScale` and `MatAXPY` for this purpose. The technical details hereof has been already discussed at full length in of this process have been discussed in depth in Siewertsen et al. (2013). Subsequent, we apply `MatMult` to every tracer of the input variable $\boldsymbol{y}_{in}$

To avoid redundant storing we do not assemble both (block diagonal) system matrices during simulation. We use the matrices provided to build just one block for each matrix type instead. The transport step is then applied as a loop over individual tracer vectors.

In contrast to the interpolation of vectors, and generally to all vector operations , each of the matrix Unlike vector interpolation and vector operations in general, each matrix operation has a significant impact on the computational time. In Section 6.2 we will present results from profiling experiments that show detailed information about showing detailed information on the time usage of each operation.

5.2 Load balancing for spatial parallelization

For spatial parallelization, the discrete tracer vectors have to be distributed to the available processes. Since biogeochemical models operate on whole water columns, profiles cannot be split without message passing. But due to the locally varying ocean depth, a tracer vector is a collection of profiles with *different* length. Thus a load balancing that takes into account only the number of profiles, but not their respective length, would be sub-optimal.

The PETSc library provides no load balancing algorithm suitable for this case. We therefore use an approach that was inspired by the idea of space filling curves presented by Zumbusch (1999).

For each profile we compute its 'computational weight', i.e. its mid, in relation to the overall computational effort, i.e. the vector length. We then project this ratio to the available number of processes, i.e we round this figure down to an integer and use the result as the index of the process the profile belongs to. By using this information the profiles can then be assigned consecutively to the processes involved.

For 0-indexed arrays this calculation is described by Algorithm 3. Its theoretical and actual performance is discussed in Section 6.3, where a comparison between Metos3D and the TMM framework is shown.

6 Results

In this section , we we will present results from our numerical experiments to verify the software. We use the introduced interface For these experiments the interface described in this paper has been used to couple the transport matrix driver with a suite of biogeochemical models. We We will also inspect the convergence behavior of both solvers included in the software. A profiling of the main parts of the algorithm complements the initial will complement the verification.

Subsequent, we perform In a second step we have performed speed-up tests to analyze the implemented load distribution load distribution implemented in our software and compare it with the TMM framework. We continue by investigating will also investigate the convergence control settings of the Newton-Krylov solver and examine the solver's behavior within parameter bounds.

6.1 Setup

We assume the PETSc environment variables are set, the toolkit is installed and the `metos3d` script is made available as a shell command.

6.0.1 Models

In order to test our interface, we couple an N, N-DOP, NP-DOP, NPZ-DOP, NPZD-DOP model hierarchy and an *original* implementation of a biogeochemical model to the transport driver. The former is implemented from scratch for this purpose. The equations are shown The experimental setup is described in Appendix B. The latter is used for the MIT General Circulation Model (cf. Marshall et al., 1997, MITgcm) biogeochemistry tutorial and described in detail in Dutkiewicz et al. (2005). We will denote it as the MITgcm-PO4-DOP model.

Generally, for every model implementation that is coupled to the transport driver via the interface a new executable must be compiled. Here, we use a convention for the directory structure to fit seamlessly into an automatic compile scheme. Within the `model` directory of the `model` repository we create a folder that is named after the biogeochemical model, i.e. `MITgcm-PO4-DOP` for instance. Within this directory we store the source code file named `model.F`A in more detail. We use this directory structure for all models. Overall,

~~while the file suffix implies a pre-processed Fortran fixed format, every programming language that is supported by the PETSc library will be accepted.~~

~~Finally, to compile all sources we invoke for instance and such create an executable named that we use for *all* the following experiments. Specific settings will be provided via option files.~~

6.0.1 ~~Data~~

~~All matrices and forcing data we use in this work are based on the example material that is freely available at (Khatiwala, 2013). This material originates from MITgcm simulations and requires post-processing. We provide the preparation scripts as well as the prepared data within the `data` repository.~~

~~The surface grid of the used domain has a longitudinal and latitudinal resolution of $2.8125°$, which results in 128×64 grid points (cf. Figure 23). Note that the Arctic has been filled in. The depth is divided into 15 vertical layers that are depicted in Table 26. This geometry translates to a (single) tracer vector length of $n_x = 52749$ and the corresponding $n_p = 4448$ profiles. Moreover, the total volume of the ocean is specified as $V \approx 1.174 \times 10^{18}\,\mathrm{m}^3$, whereas the minimal and maximal volume of a grid box is $V_{\min} \approx 8.357 \times 10^{11}\,\mathrm{m}^3$ and $V_{\max} \approx 6.744 \times 10^{13}\,\mathrm{m}^3$, respectively. The temporal resolution is at $\Delta t = 1/2880$, which is equivalent to an (ocean) time step of 3 hours assuming that a year consists of 360 days.~~

~~The computation of the photosynthetically available short wave radiation is the same for all models. It is deduced from the insolation, which is computed on the fly using the formula of Paltridge and Platt (1976). Here, for the topmost layer latitude and ice cover data is required, i.e. $n_b = 2$. For the former we use a single latitude file, i.e. $n_{b,1} = 1$, and for the latter twelve ice cover files, $n_{b,2} = 12$.~~

~~Additionally, the depths and heights of the vertical layers are required, i.e. $n_d = 2$ domain data sets. Each consist of only one file, i.e. $n_{d,1} = 1$ and $n_{d,2} = 1$. The information is used to compute the attenuation of light by water, to determine the fluxes of particulate organic phosphorus and to approximate a derivative with respect to depth. Note that the order in which the data sets are provided is important and must correspond to the order used within the model implementation. Moreover, as previously mentioned, twelve implicit transport matrices, i.e. $n_{imp} = 12$, and twelve explicit transport matrices, i.e. $n_{exp} = 12$ are provided. We always start a simulation at $t_0 = 0$ and perform $n_t = 2880$ iterations per model year.~~

6.1 Solver

We begin our verification by computing a steady annual cycle for every model~~with~~ , using both solvers. ~~Regarding~~ When using the spin-up ~~,~~ we set no tolerance and let the solver iterate for 10,000 model years. The Newton approach is set to a line search variant and the Krylov subspace solver to GMRES. All other settings are left ~~to default, in particular the~~ at default, so overall absolute tolerance is at 10^{-8} and the maximum number of inner iterations is 10,000.

The parameter values ~~we use~~ used for the MITgcm-PO4-DOP model are ~~depicted~~ listed in Table 27 ~~and named~~ under the heading $\boldsymbol{u}_d$ ~~therein~~. Table 28 ~~depicts~~ lists the parameter values used for the N, N-DOP, NP-DOP, NPZ-DOP, NPZD-DOP model hierarchy. If not stated otherwise the initial value is set to $2.17\ \mathrm{mmol\,P\,m^{-3}}$ for N or PO4 and $0.0001\ \mathrm{mmol\,P\,m^{-3}}$ for ~~the~~ all other tracers.

~~For the MITgcm-PO4-DOP model a comparison of the~~ A comparison of convergence towards a steady annual cycle for both solvers, applied to the MITgcm-PO4-DOP model, is shown in Figure 24. We observe that ~~the solutions converge to the same difference in~~ both solvers reach the same difference between consecutive iterations at the end. ~~Moreover,~~ Table 29 shows the ~~difference~~ differences between both solutions in Euclidean ~~norm.~~ and volume-weighted norms, cf. Eq. (4). ~~Additionally,~~ Figure 25 depicts the difference between both solutions for one tracer at the surface layer. Except for ~~the numerical error ,~~ numerical error both solvers obviously compute the same solution.

Figures 26 and 27 show the convergence behavior of both solvers for the N ~~respectively~~ and the N-DOP model, respectively. ~~There is no essential difference in comparison to the MITgcm-PO4-DOP model~~Again both solvers end with approximately the same accuracy and produce similar results. ~~An inspection of the surface~~ This impression is confirmed by an inspection of Figures 28 and 29 ~~confirms this impression. There is no peculiarity shown in~~ as well as Table 29~~either~~.

However, ~~for the NP-DOP model~~ in Figure 210 ~~shows~~ a different behavior ~~of~~ can be observed for the Newton-Krylov solver at the end of the solution process, applied to the NP-DOP model. ~~A closer~~ Closer inspection reveals a peak every 30 model years, which ~~obviously~~ results from the settings of the inner solver, where GMRES is set to perform a restart every 30 years~~by default~~. ~~Surface~~ This option is chosen to reduce the internal storage requirement, but may lead to stagnation for indefinite matrices, cf. Saad (2003, Sect. 6.5.6). It is likely that the Jacobian at some Newton step becomes indefinite, and thus we assume that this is the case here. Figure 211 and Table 29 ~~, however,~~ do not indicate any ~~effect~~ influence on the solution, however.

~~The~~ For the NPZ-DOP ~~and~~ or the NPZD-DOP ~~models show a different behavior regarding~~ model the Newton solver shows a different behavior. For both models ~~,~~ the solver does not converge ~~with default settings as shown~~ if default settings are used, as depicted in Figure 212 (top) and Figure 213 (top). ~~It can be seen that the reduction~~ Reduction of the residual per step is quite low, which results in a huge number of iterations. ~~Here,~~ In this case the solver was stopped after 50 iterations (the default setting), which ~~already~~

~~is a high number~~ is quite high for Newton's method. ~~The~~ ~~reason is~~ This behavior was caused by the fact that convergence of ~~the~~ this method – even in its so-called globalized or damped version used here – ~~still may depend~~ at times still depends on the initial guess $\boldsymbol{y}^0$. We therefore used a different one, which was successful ~~for~~ with the NPZD-DOP model, see Figure 213 (middle). ~~For~~ With the NPZ-DOP model, ~~it still was not~~ this procedure still did not work, see Figure 212 (middle).

However, the result of a second and much easier way to achieve convergence can be ~~deduced already from~~ seen in Figure 212 (top) and Figure 213 (top). ~~The stopping criterion of the inner iterations of the Newton solver is less restrictive if the~~ If the last Newton iteration ~~was not very successful, which is~~ step did not lead to a big reduction of the residual, which was obviously the case here, the stopping criterion (8) for the inner iterations of the Newton solver becomes less restrictive. ~~The~~ If this criterion is sharpened the number of inner iterations increases and thus the accuracy of the Newton direction ~~is improved when the inner criterion is sharpened, thus somehow contradicting the~~ improve. This somewhat contradicts the idea formulated in Eisenstat and Walker (1996). ~~This can be easily~~ Sharpening can easily be achieved by decreasing γ, ~~here~~ in this case to $\gamma = 0.3$. This tuning ~~now~~ led to convergence, see Figure 212 (bottom) and Figure 213 (bottom). ~~With this settings , the respective solutions are the same as~~ When using these settings the ~~ones obtained by the~~ same solutions are obtained as with the spin-up, ~~when~~ if numerical errors are neglected (see Figures 214 and 215). This ~~is also~~ result is confirmed by evaluating the differences in the norm, see Table 29.

~~Overall, we observe that~~ It can be observed that as a rule the Newton-Krylov solver does not reach ~~the default tolerance~~ default tolerance within the last Newton step and iterates unnecessarily for 10,000 model years~~within the last Newton step~~. ~~Thus, we~~ From now on we will therefore limit the inner Krylov iterations to ~~200 in the following experiments.~~ 200. ~~Moreover, for further investigations with~~

For our next investigations using the MITgcm-PO4-DOP model we ~~change~~ will alter the convergence settings as well to get rid of the over-solving ~~that we observe at the beginning~~observed before. ~~Referring to this, more detailed experiments~~ More detailed experiments on this subject are presented in Section 6.4.

6.2 Profiling

~~In following~~ In the next two sections we ~~investigate~~ will investigate more closely some technical aspects of the implementation~~more closely~~. ~~First of all, we are interested in~~ We will first look at the distribution of ~~the~~ computational time among the main operations of ~~a~~ one model year.

For this ~~, purpose~~ we perform a *profiled* sequential run for each model~~at which we iterate~~ , iterating for 10 model years. ~~The analysis of the~~ An analysis of our profiling results is shown in Figures 216 - 218~~- 216~~. ~~Regarding~~ When using the MITgcm-PO4-DOP model, for instance, ~~we observe that~~ the biogeochemical model takes up 40% of ~~the~~ computational time. ~~The interpolation~~ Interpolation of matrices (`MatCopy`, `MatScale` and `MatAXPY`) amounts to approximately ~~a~~ one third. ~~The matrix~~ Matrix vector multiplication (`MatMult`) takes up a quarter of ~~the~~ all computations and all other operations amount to 0.5%.

~~Moreover, we recognize that the more tracers are involvedthe more the~~ Our data also suggest that the greater the number of tracers involved, the more dominant matrix vector multiplication becomes~~dominant~~. ~~For the N model it~~ The `MatMult` operation takes up $19,8\%$ of ~~the computational time , whereas~~ computational time for the N model, but $56,7\%$ for the NPZD-DOP model~~the `MatMult` operation amounts to $56,7\%$~~. The ~~possible implications~~ implications of these results are discussed in Section 7. Additionally, in Table 210 the absolute timings and the computing time per tracer versus number of tracers are shown.

~~This profiling capability was also used as the software was ported by Siewertsen et al. (cf. 2013)~~ Siewertsen et al. (cf. 2013) also made use of this profiling capacity when porting the software to an NVIDIA graphics processing unit (GPU). The authors investigated the impact of the accelerator's hardware on the simulation of biogeochemical models. ~~The~~ Their work comprises a detailed discussion ~~on peak performance as well as~~ of peak performance and memory bandwidth and includes a counting of floating point operations.

6.3 Speed-up

In this section ~~, we investigate~~ we will investigate in detail the performance of the load balancing algorithm ~~in detail and compare the~~ and compare our results with the ~~parallel performance of the TMM~~ scalability provided by the TMM framework. We compile both drivers ~~with~~ using the same biogeochemical model. ~~For this purpose we choose~~ We choose the MITgcm-PO4-DOP ~~since it is part of the TMM as well and, consequently, we have the same setup.~~ model using the same time step, initial condition as well as boundary and domain data.

~~We run the tests on a hardware that~~ Our tests are run on hardware located at the computing center of Kiel University~~. It is~~ : an Intel® Sandy Bridge EP architecture with Intel Xeon® E5-2670 CPUs that consist of 16 cores running at 2.6 GHz. ~~Regarding our implementation we~~ We perform 10 tests for our implementation, using 1 to 256 cores.

Each test consists of a simulation run of three model years, ~~at which~~ where each year is timed separately. For the TMM framework we use 1 to 192 cores and run 5 tests on each core. ~~Here, we~~ We use the given output ~~, which is~~ here, which shows the timing for ~~the~~ one whole run.

~~Overall, for the calculation of the~~ To calculate speed-up and efficiency ~~results~~ we use the minimum timings for a specific number of cores. ~~Moreover, all~~ All timings are related to the timing of a sequential run. For a set of ~~measured~~ computational times $(t_i)_{i=1}^{N}$ measured during our experiments, with $N = 192$ or $N = 256$~~we calculate the speedup~~, respectively, we calculate speed-up as $s_i = t_1/t_i$ and ~~the~~ efficiency as $e_i = 100 * s_i/i$.

~~Additionally, referring to the implemented load distribution~~ To investigate the load distribution implemented by us (cf. Section 5.2) ~~,~~ we compute the best ~~possible ratio~~ ratio possible between a sequential and a parallel run. ~~For all number~~ Using Algorithm 3 we first compute the load distribution for all numbers of processes, i.e. $i = 1, \ldots, 260$, ~~we compute the load distribution using Algorithm 3 and~~ and then retrieve the maximum (local) length $n_{i,max}$. ~~For the~~ To calculate speed-up we divide the vector length by this value, i.e. $s_i = n_y/n_{i,max}$, and ~~for the~~ to calculate efficiency we again ~~calculate~~ use $e_i = 100 * s_i/i$.

Figure 219 depicts ~~the~~ ideal, theoretical and actual ~~speedup respectively efficiency.~~ data for speed-up and efficiency. ~~Regarding the implemented load distribution~~ Here, the term 'ideal' refers to a perfectly parallelizable program and a perfect hardware with no delay on memory access or communication. Regarding the load distribution implemented by us a good (theoretical) performance can be observed over the whole range of processes~~can be observed~~. ~~Moreover, we recognize~~ This refers again to a perfect hardware except that we distribute a collection of profiles of different length here.

The data also show that a parallel run of Metos3D on the Intel hardware ~~reaches~~ achieves close to perfect performance when using between 100 and 140 cores~~almost best performance~~. ~~In this range the efficiency is~~ Efficiency is at about 95% ~~and the~~ in this range and speed-up nearly corresponds to the number of processes. ~~Indeed, the~~ In fact speed-up ~~still rises~~ may rise still further up to slightly over 160~~but requires at least~~ , but a minimum of 200 processes ~~to reach thisfactor~~are required to achieve this.

In ~~contrast, the performance~~ comparison, the scalability of the TMM ~~is poor~~framework is not optimal. ~~The efficiency drops from the beginning and a speedup higher than 40 is never reached.~~ Efficiency drops off immediately and speed-up never rises above 40. ~~From~~ For 120 cores ~~up~~ and above Metos3D is at least 4 times faster. ~~Interestingly, there is~~ Interestingly, for low numbers of processes a significant drop in performance ~~at the beginning~~ can be observed for both drivers. ~~The possible implications are shortly discussed~~ The implications of this are discussed briefly in Section 7. ~~However,~~ We did not investigate this effect any further, however, since the results ~~give us a good orientation anyway this effect is not investigated further.~~ presented here already provide a good guideline.

6.4 Convergence control

After ~~a~~ this basic verification and ~~a review of~~ the review of some technical aspects of our implementation, we ~~investigate the settings to control the~~ will now investigate those settings that control convergence of the Newton-Krylov solver. ~~Again, we use~~ Once again we use only the MITgcm-PO4-DOP model~~only~~. Our intention here is to eliminate the oversolving ~~that we observe~~ we observed during the first 200 iterations as shown in Figure 24. This effect occurs ~~,~~ if the accuracy of the inner solver is significantly higher than the resulting Newton residual (cf. Eisenstat and Walker, 1996). The relation between ~~those~~ these two is controlled by the parameters γ and the α ~~parameter depicted~~ used in Equation (8).

~~Hence,~~ To investigate the influence of these parameters on convergence we compute the reference solution ~~from~~ described in Section 6.1 ~~with~~ using different values of γ and α~~to investigate their influence on the convergence behavior~~. We set ~~the~~ overall tolerance to the ~~measured difference of~~ difference measured between consecutive states after 3,000 model years of spin-up, i.e. approximately 9.0×10^{-4}. ~~We let the value of~~ γ ~~vary~~ is varied from 0.5 to 1.0 in steps of 0.1 and α ~~is chosen~~ from 1.1 to 1.6, also in steps of 0.1~~as well~~. This ~~is~~ makes for a total of 36 model evaluations.

Figure 220 depicts the ~~required~~ number of model years and Newton steps required as a function of γ and α. We observe that the overall number of years decreases ~~, as both parameters tend to~~ as the two parameters tend towards 1.0 and 1.1, respectively. In contrast, the number of Newton steps increases, i.e. the Newton residual is computed more often and the inner steps become shorter.

Consequently, since the computation of ~~a~~ one residual is negligible in comparison to the simulation of ~~a~~ one model year, we focus on decreasing the overall number of model years. A detailed inspection of the results reveals that for $\gamma = 1.0$ and $\alpha = 1.2$ the solver reaches the ~~set tolerance~~ tolerance set above after approximately 450 model years, which is significantly less than the 600 ~~if~~ years needed when using the default settings. ~~Thus, we~~

We therefore use these values for ~~the~~ our next experiment.

6.5 Parameter samples

~~Until now we solved the given~~ So far we have solved the model equations for one (reference) ~~parameter set~~ set of parameters only. During ~~an optimizationa solution~~ optimization, however, solutions must be computed for various parameter sets. ~~Thus, we perform the next experiments in order to study~~ Our next experiments therefore investigate the solver's behavior with regard to ~~other~~ different model parameters. ~~Again, we~~ Once again we only use the MITgcm-PO4-DOP model~~only~~. ~~For this purpose, using~~ Using the MATLAB® routine `lhsdesign`, we create 100 Latin Hypercube (cf. McKay et al., 1979) samples within the bounds

~~that are depicted~~ described in Table 27. ~~We set the overall tolerance again~~ As before we set overall tolerance to a value ~~that is comparable with~~ comparable to 3,000 spin-up iterations and let the Newton solver compute a solution for each parameter sample.

Figure 221 shows histograms of the total number of model years ~~respectively~~ or Newton steps required to solve the model equations. We observe that most computations converge ~~in between~~ after 400 to 550 model years and require 10 to 30 Newton steps. Interestingly, ~~regarding the latter~~ there is a high peak around 15 and a smaller ~~peak around 12.~~ one around 12 for the Newton method. ~~Moreover, we recognize~~ We also find some outliers in both graphs. Nevertheless ~~, all started model evaluation~~ all model evaluations we started converged towards a solution within the desired tolerance.

7 Conclusions

We designed and implemented a simulation framework for the computation of steady annual cycles for a ~~general~~ generalized class of marine ecosystem models in 3-D, driven by ~~pre-computed transport matrices~~ transport matrices pre-computed in an off-line mode. ~~The~~ Our framework allows computation of ~~the steady cycle(s) by a~~ steady cycles by spin-up or by a globalized Newton method. The software ~~is completely realized as (or using available)~~ has been realized as open source code throughout.

We also introduced a software interface for water column-based biogeochemical models. ~~On one hand, we showed~~ We demonstrated the applicability and flexibility of this interface by coupling the biogeochemical component used in the MITgcm general circulation model to the simulation framework. ~~On the other hand, we coupled~~ To test the general usability of the interface we then coupled our own implementations of five ~~other biogeochemical models (also~~ different biogeochemical models of varying complexity (already used in Kriest et al. (2010)) ~~with different complexity to show the interface's generality~~to the framework. ~~Their source code~~ The source code of these models is also available ~~within the software~~ as part of the software package, and may serve as ~~templates for~~ template for the implementation or adaption of other models.

We implemented a transient solver based on the transport matrix approach, where all matrix operations and ~~the evaluation of the~~ evaluations of biogeochemical models are performed ~~with~~ by spatial parallelization via MPI using the PETSc library. The ~~needed transport matrices are directly available~~ transport matrices needed for this process are available directly and require no pre-processing.

We realized both a spin-up (or fixed-point iteration) and a globalized Newton solver for the computation of steady cycles. We compared ~~these~~ the performance of both solvers and made the following observations: Both ~~deliver~~ delivered the same results (up to a reasonable precision) on convergence. The spin-up ~~converges with~~ converged when using standard sets of parameters, which were taken from Kriest et al. (2010), ~~for~~ and equally distributed values for ~~each tracer~~all tracers. The Newton solver ~~showed the same behavior~~ did the same for the four models of lower complexity. ~~For the other two, it~~ It did not converge ~~with the standard setting of its parameters and the mentioned~~ for the other two models when using standard parameter settings and an initial distribution of tracers as described above. For both of these ~~two~~ more complex models ~~, convergence was~~ convergence could be achieved by increasing the number of inner iterations in the Newton solver, which is realized by decreasing the parameter γ in (8). For one of these models ~~, the same could~~ convergence could also be achieved by choosing a different initial guess.

~~Concerning~~ With regard to performance, the Newton solver was about 6 times faster for all models. It can be concluded that for complex models the Newton method requires more ~~thorough solver parameter setting for complex models~~attention to solver parameter settings, but then is superior ~~in any case~~to the spin-up, at least ~~for the considered parameter sets~~ when using parameter sets as described above.

~~We studied the dependency~~ In a next step we investigated how performance of the Newton ~~performance with respect to~~ method is influenced by the two solver parameters α, γ in (8)~~for one exemplary model~~ , using one model as an example. ~~With an~~ Employing the optimal choice derived from these experiments (~~for~~ and one model parameter set), we then ~~investigated the dependency of the needed~~ studied the number of Newton iterations and overall model years needed for 100 latin hypercube model parameter samples. This ~~test is important~~ is an important test for the usability of the Newton method ~~for example in a optimization run where~~ in various kinds of optimization runs, for example if model parameters are varied by the optimizer. ~~It turned out that there is a~~ As it turned out there was a certain variance in the ~~needed steps and thus~~ number of steps needed and thus in the overall effort, but ~~that there are~~ there were no extreme outliers. ~~We conclude~~ Our conclusion is that the Newton method ~~–~~ is appropriate for optimization, at least for this model ~~– is appropriate for optimization~~, and faster than the usually robust spin-up.

We further analyzed ~~the proportions in time that the different pieces of the simulation in~~ which proportion of computational time is utilized by different parts of our software during simulation of one model year~~need~~. ~~It turned out that , with increasing~~ Our experiments showed that with an increase in the number of tracers ~~,~~ the matrix-vector operations ~~dominate and thus have the most~~ started to dominate the process, thus offering the greatest potential for further performance tuning. This ~~is despite the fact that~~ was the case even though the transport operator was the same for every tracer~~is the same~~. ~~However, it still has to be evaluated, whose effort is proportional to the number of tracers in the model. In contrary, the biogeochemical interactions~~ In contrast all

biogeochemical interactions contained in the nonlinear coupling terms $\boldsymbol{q}_j$, which ~~are mostly~~ mostly are spatially local, become less performance-relevant as the number of tracers increases.

Finally, we implemented a load balancing ~~that exploits the different depths of the~~ mechanism which exploits the fact that water columns in the ocean ~~that result in different lengths of the corresponding data vectors~~ vary in depth, resulting in data vectors of variable length. ~~With this balancing , a nearly~~ Using this balancing method a close to optimal speed-up by spatial parallelization ~~up to about a comparably~~ was achieved up to the relatively high number of 128 processes~~was possible~~. ~~This is a huge difference to the performance with~~ The difference to standard load balancing is immense.

~~Summarizing, the presented software framework is an appropriate tool to be used in parameter optimization and model assessment runs. It has~~ To summarize, the software framework presented here offers high flexibility w.r.t. models and steady cycle solvers~~, offers improved parallel performance and can be easily combined with any optimization method~~. The ~~option for effective high spatial parallelization allows the use of gradient based optimization methods, since they are – in contrast to evolutionary algorithms – less parallelizable~~implemented load balancing scheme results in significant improvement in parallel performance. ~~Our results show that the parallelization effort is well-invested in the simulation itself~~Especially, the apllied Newton solver can be tuned to converge for all six biogeochemical models.

8 Code availability

Name of software: Metos3D (Simulation Package v0.3.2)
Developer: Jaroslaw Piwonski
Year first available: 2012
Software required: PETSc 3.3
Program language: C, C++, Fortran
Size of installation: 1.6 GB
Availability and ~~Cost~~costs: free software, GPLv3
Software homepage: https://metos3d.github.com/metos3d

The toolkit is maintained using the distributed revision control system `git`. All source codes are available at GitHub (https://github.com). The current versions of `simpack` and `model` are tagged as `v0.3.2`. The data ~~is repository is at~~ repository is tagged as version `v0.2`. All experiments presented in this work were carried out using ~~this~~ these versions. ~~The associated~~ Associated material is stored in the `2016-GMD-Metos3D` repository.

To install the software ~~, the user~~ users should visit the homepage and follow ~~the~~ instructions. ~~Whereas in the future an installation will always reflect the current~~ Future installations will reflect the state of development ~~, the user can always invoke~~ at that point of time, but users may still retrieve the versions used in this work by invoking `git checkout v0.3.2` in the `simpack` and `model` repository as well as `git checkout v0.2` in the `data` ~~repository to retrieve the versions used in this work.~~ repositories.

Appendix A: Experimental setup

We assume that all PETSc environment variables have been set, the toolkit has been installed and the `metos3d` script has been made available as a shell command.

A1 Models

In order to test our interface we couple an N, N-DOP, NP-DOP, NPZ-DOP, NPZD-DOP model hierarchy as well as an implementation of Dutkiewicz et al. (2005)'s original biogeochemical model. The former has been implemented from scratch for this purpose. The corresponding equations are shown in Appendix B. The latter is the model used for the MIT General Circulation Model (cf. Marshall et al., 1997, MITgcm) biogeochemistry tutorial. We will denote it as the MITgcm-PO4-DOP model.

For every model implementation that is coupled to the transport driver via the interface a new executable must be compiled. We have established naming conventions for the directory structure so that it fits seamlessly into an automatic compile scheme. We create a folder that is named after the biogeochemical model, for instance `MITgcm-PO4-DOP`, within the `model` directory of the `model` repository.

Within this folder the source code file named `model.F` is stored. This directory structure is used for all models. Although the file suffix used here implies a pre-processed Fortran fixed format, any programming language supported by the PETSc library will be accepted.

To compile all sources (still using the same example) we invoke

```
$> metos3d simpack MITgcm-PO4-DOP
```

and obtain an executable named

```
metos3d-simpack-MITgcm-PO4-DOP.exe
```

which we will use for *all* experiments described below. Specific settings will be provided via option files.

A2 Data

All matrices and forcing data used in this work are based on the example material available at (Khatiwala, 2013). This material originates from MITgcm simulations and requires some post-processing. The corresponding preparation scripts are provided along with the processed data in the `data` repository.

The surface grid of the domain used has a longitudinal and latitudinal resolution of 2.8125°, which produces 128×64 grid points (cf. Figure 23). Note that the Arctic has been filled in, i.e. set to land. This originates in the data provided at the TMM webpage (cf. Khatiwala, 2013). The depth is divided into 15 vertical layers as described in Table 26. This geometry translates to a (single) tracer vector length of $n_x = 52749$ and to $n_p = 4448$ corresponding profiles. Temporal resolution is at $\Delta t = 1/2880$, which is equivalent to an (ocean) time step of 3 hours, assuming that one year consists of 360 days.

The method of computing photosynthetically available short wave radiation is the same for all models. It is deduced from insolation, which is computed on the fly using the formula of Paltridge and Platt (1976). For this purpose latitude and ice cover data are required for the topmost layer, i.e. $n_b = 2$. We use a single latitude file for the former, i.e. $n_{b,1} = 1$, and twelve ice cover files for the latter, $n_{b,2} = 12$.

The depths and heights of all vertical layers are required as well, so we have $n_d = 2$ domain data sets. Each set consists of only one file, i.e. $n_{d,1} = 1$ and $n_{d,2} = 1$. This information is used to compute the attenuation of light by water to determine the fluxes of particulate organic phosphorus and to approximate a derivative with respect to depth. Note that these data sets have to be provided in a specific order, which must correspond to the order used within the model implementation. In addition, twelve implicit transport matrices, i.e. $n_{imp} = 12$, and twelve explicit transport matrices, i.e. $n_{exp} = 12$, are provided as mentioned previously. Each simulation starts at $t_0 = 0$ and performs $n_t = 2880$ iterations per model year.

Appendix B: Model equations

The ~~here presented~~ N, N-DOP, NP-DOP, NPZ-DOP and NPZD-DOP model hierarchy presented here is based on the descriptions used by Kriest et al. (2010). ~~The introduced parameters~~ All parameters introduced are shown in Table 28.

B1 Short wave radiation

As mentioned in Section A2, ~~the~~ short wave radiation for the *topmost* layer is deduced from ~~the insolationthat~~ insolation, which is computed on the fly using the formula of Paltridge and Platt (1976). ~~Here,~~ For this purpose latitude ϕ and ice cover σ_{ice} data ~~is~~ are required. We denote the computed value by $I_{SWR} = I_{SWR}(\phi, \sigma_{ice})$. For ~~the lower layers their depths~~ all lower layers data on depth $(z_j)_{j=1}^{n_x}$ and ~~heights~~ height $(dz_j)_{j=1}^{n_x}$ are required. ~~Additionally, the attenuation of~~ Attenuation by water is described by the coefficient k_w ~~respectively the attenuation of~~ and attenuation by phytoplankton (chlorophyll) by k_c.

B1.1 Implicit phytoplankton

~~For the N and the~~ For models N and N-DOP ~~model the~~ short wave radiation is computed *without* phytoplankton, i.e.

$$I_j = I_{SWR} \begin{cases} I'_j & j = 1 \\ I'_j \prod_{k=1}^{j-1} I_k & \text{else} \end{cases}$$

where $I'_j = \exp(-k_w\, dz_j/2)$, ~~$I_k = \exp(-k_w\, dz_k)$~~ $I_j = \exp(-k_w\, dz_j)$, and j is the ~~actual layer index~~ index of the individual layers.

B1.2 Explicit phytoplankton

~~For the~~ For models NP-DOP, NPZ-DOP and NPZD-DOP ~~model the~~ short wave radiation is computed *with* phytoplankton included, i.e.

$$I_{P,j} = I_{SWR} \begin{cases} I'_{P,j} & j = 1 \\ I'_{P,j} \prod_{k=1}^{j-1} I_{P,k} & \text{else} \end{cases}$$

where $I'_{P,j} = \exp(-(k_w + k_c\, y_{P,j})\, dz_j/2)$ and $I'_{P,k} = \exp(-(k_w + k_c\, y_{P,k})\, dz_k)$.

B2 N model

The simplest model used here consists of nutrients (N) only, i.e. $\boldsymbol{y} = (\boldsymbol{y}_N)$. The equation is presented in Table B1 ~~depicts the equation~~. ~~The biological~~ Biological uptake is computed as

$$f_P(\boldsymbol{y}_N, I) = \mu_P\, y_P^* \frac{\boldsymbol{y}_N}{K_N + \boldsymbol{y}_N} \frac{I}{K_I + I},$$

where ~~phytoplankton is *implicitly* set to $y_P^* = 0.0028$ mmol P/m^3~~ the implicitly prescribed concentration of phytoplankton is set to $y_P^* = 0.0028\,\mathrm{mmol\,P\,m^{-3}}$. Note that y_P^* could be a free model parameter as well. However, we stick to this formulation to be consistent with Kriest et al. (2010). The N model introduces $n_u = 5$ parameters, ~~where~~ with $\boldsymbol{u} = (k_w, \mu_P, K_N, K_I, b)$.

B3 N-DOP model

The N-DOP model consists of nutrients (N) and dissolved organic ~~phosphorous~~ phosphorus (DOP), i.e. $\boldsymbol{y} = (\boldsymbol{y}_N, \boldsymbol{y}_{DOP})$. ~~The computation of the~~ Computation of biological uptake remains the same. The equations are shown in Table B2 ~~depicts the equations~~. The N-DOP model introduces $n_u = 7$ parameters, ~~where~~ with $\boldsymbol{u} = (k_w, \mu_P, K_N, K_I, \sigma_{DOP}, \lambda_{DOP}, b)$.

B4 NP-DOP model

The NP-DOP model consists of nutrients (N), phytoplankton (P), and dissolved organic ~~phosphorous~~ phosphorus (DOP), i.e. $\boldsymbol{y} = (\boldsymbol{y}_N, \boldsymbol{y}_P, \boldsymbol{y}_{DOP})$. ~~Here , the~~ Here nutrient uptake by

(explicit) phytoplankton is computed as

$$f_P(\boldsymbol{y}_N, \boldsymbol{y}_P, I_P) = \mu_P \boldsymbol{y}_P \frac{\boldsymbol{y}_N}{K_N + \boldsymbol{y}_N} \frac{I_P}{K_I + I_P}.$$

~~The computation~~ Computation of short wave radiation ~~changes~~ is altered as well (see Section B1.2). ~~Additionally,~~ In addition a quadratic loss term for phytoplankton is introduced~~and~~ , as is a grazing function

$$f_Z(\boldsymbol{y}_P) = \mu_Z y_Z^* \frac{\boldsymbol{y}_P^2}{K_P^2 + \boldsymbol{y}_P^2},$$

where ~~zooplankton is implicitly set to $y_Z^* = 0.01$ mmol P/m^3~~the implicitly prescribed concentration of zooplankton is set to $y_Z^* = 0.01\ \mathrm{mmol\,P\,m^{-3}}$. Again, we stick to this formulation to be consistent with Kriest et al. (2010), though y_Z^* could be a free model parameter. The equations are shown in Table B3~~depicts the equations~~. The NP-DOP model introduces $n_u = 13$ parameters, ~~where~~ with $\boldsymbol{u} = (k_w, k_c, \mu_P, \mu_Z, K_N, K_P, K_I, \sigma_{DOP}, \lambda_P, \kappa_P, \lambda'_P, \lambda_{DOP}, b)$.

B5 NPZ-DOP model

The NPZ-DOP model consists of nutrients (N), phytoplankton (P) zooplankton (Z) and dissolved organic ~~phosphorous~~ phosphorus (DOP), i.e. $\boldsymbol{y} = (\boldsymbol{y}_N, \boldsymbol{y}_P, \boldsymbol{y}_Z, \boldsymbol{y}_{DOP})$. The production function remains the same. ~~The~~ For the computation of grazing~~takes explicit zooplankton into account~~, zooplankton is dealt with explicitly, i.e.

$$f_Z(\boldsymbol{y}_P, \boldsymbol{y}_Z) = \mu_P \boldsymbol{y}_Z \frac{\boldsymbol{y}_P^2}{K_P^2 + \boldsymbol{y}_P^2}.$$

The equations are shown in Table B4~~depicts the equations~~. The NPZ-DOP model introduces $n_u = 16$ parameters, ~~where~~ with $\boldsymbol{u} = (k_w, k_c, \mu_P, \mu_Z, K_N, K_P, K_I, \sigma_Z, \sigma_{DOP}, \lambda_P, \lambda_Z, \kappa_Z, \lambda'_P, \lambda_Z, \lambda DOP, b)$. $\lambda'_P, \lambda'_Z, \lambda'_{DOP}, b)$.

B6 NPZD-DOP model

~~The NPZ-DOP~~ The NPZD-DOP model consists of nutrients (N), phytoplankton (P) zooplankton (Z), detritus (D) and dissolved organic ~~phosphorous~~ phosphorus (DOP), i.e. $\boldsymbol{y} = (\boldsymbol{y}_N, \boldsymbol{y}_P, \boldsymbol{y}_Z, \boldsymbol{y}_D, \boldsymbol{y}_{DOP})$. ~~The equations mainly remains the same, except a depth dependent~~ Most equations are unchanged, except that a depth-dependent linear sinking speed is introduced for detritus. The equations are shown in Table B5~~depicts the equations~~. The NPZD-DOP model introduces $n_u = 16$ parameters, ~~where $\boldsymbol{u} = (k_w, k_c, \mu_P, \mu_Z, K_N, K_P, K_I, \sigma_Z, \sigma_{DOP}, \lambda_P, \lambda_Z, \kappa_Z, \lambda_P, \lambda_Z, \lambda_D, \lambda_{DOP}, a_D, b_D)$.~~ with $\boldsymbol{u} = (k_w, k_c, \mu_P, \mu_Z, K_N, K_P, K_I, \sigma_Z, \sigma_{DOP}, \lambda_P, \lambda_Z, \kappa_Z, \lambda'_P, \lambda'_Z, \lambda'_D, \lambda'_{DOP}, a_D, b_D)$.

~~.5mm~~

Acknowledgements. The authors would like to thank S. Khatiwala for providing support on the subject of transport matrices and for ~~providing~~ offering the whole TMM material freely on the internet. Furthermore, both authors would like to thank I. Kriest and A. Oschlies for many fruitful discussions. In particular, Jaroslaw Piwonski would like to thank I. Kriest for teaching him patiently so much about biogeochemical models. This work was partly funded by ~~*The Future Ocean* cluster .~~ the DFG cluster *The Future Ocean*, grant ??? and by the German Federal Ministry of Education and Research (BMBF), grant 01LP1512B. The sole responsibility for the report's content lies with the authors. Finally the authors would like to thank the editor and two anonymous referees for their valuable comments.

References

Balay, S., Gropp, W. D., McInnes, L. C., and Smith, B. F.: Efficient Management of Parallelism in Object Oriented Numerical Software Libraries, in: Modern Software Tools in Scientific Computing, edited by Arge, E., Bruaset, A. M., and Langtangen, H. P., pp. 163–202, Birkhäuser Press, Basel, 1997.

Balay, S., Brown, J., , Buschelman, K., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Smith, B. F., and Zhang, H.: PETSc Users Manual, Tech. Rep. ANL-95/11 - Revision 3.3, Argonne National Laboratory, Lemont, 2012a.

Balay, S., Buschelman, K., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Smith, B. F., and Zhang, H.: PETSc Web page, http://www.mcs.anl.gov/petsc/ (last access: 12 July 2013), 2012b.

Bernsen, E., Dijkstra, H. A., and Wubs, F. W.: A method to reduce the spin-up time of ocean models, Ocean Modelling, 20, 380 – 392, doi:10.1016/j.ocemod.2007.10.008, 2008.

Bryan, K.: Accelerating the Convergence to Equilibrium of Ocean-Climate Models, Journal of Physical Oceanography, 14, 666–673, doi:10.1175/1520-0485(1984)014<0666:ATCTEO>2.0.CO;2, 1984.

Ciric, L. B.: A Generalization of Banach's Contraction Principle, Proceedings of the American Mathematical Society, 45, 267–273, 1974.

Danabasoglu, G., McWilliams, J. C., and Large, W. G.: Approach to Equilibrium in Accelerated Global Oceanic Models, Journal of Climate, 9, 1092–1110, doi:10.1175/1520-0442(1996)009<1092:ATEIAG>2.0.CO;2, 1996.

Dennis, J. and Schnabel, R.: Numerical methods for unconstrained optimization and nonlinear equations, Society for Industrial and Applied Mathematics, Philadelphia, 1996.

Dutkiewicz, S., Sokolov, A. P., Scott, J., and Stone, P. H.: A three-dimensional ocean-seaice-carbon cycle model and its coupling to a two-dimensional atmospheric model: Uses in climate change studies, Tech. Rep. 122, MIT Joint Program on the Science and Policy of Global Change, Cambridge, 2005.

Eisenstat, S. C. and Walker, H. F.: Choosing the Forcing Terms in an Inexact Newton Method, SIAM Journal on Scientific Computing, 17, 16–32, doi:10.1137/0917003, 1996.

Evans, G. T. and Garçon, V. C.: One-Dimensional Models of Water Column Biogeochemistry, Report of a workshop held in Toulouse, France, November-December 1995. GOFS Report N°23/97, JGOFS Bergen, Norway, 1997.

Table B1. Equations for the N model with $f_P = f_P(y_N, I)$ and $E_j = f_P\, dz_j$.

	Euphotic zone	Sinking
$q_N(y) =$	$-f_P$	$+E_j\, \partial_z (z/z_j)^{-b}$

Table B2. Equations for the N-DOP model with $f_P = f_P(y_N, I)$ and $E_j = \bar{\sigma}_{DOP}\, f_P\, dz_j$.

	Euphotic zone	All layers	Sinking
$q_N(y) =$	$-f_P$	$+\lambda'_{DOP}\, y_{DOP}$	$+E_j\, \partial_z (z/z_j)^{-b}$
$q_{DOP}(y) =$	$+\sigma_{DOP} f_P$	$-\lambda'_{DOP}\, y_{DOP}$	

Table B3. Equations for the NP-DOP model with $f_P = f_P(y_N, y_P, I_P)$, $f_Z = f_Z(y_P)$ and $E_j = \bar{\sigma}_{DOP}\, f_Z\, dz_j$.

	Euphotic zone				All layers		Sinking
$q_N(y) =$	$-f_P$					$+\lambda'_{DOP}\, y_{DOP}$	$+E_j\, \partial_z (z/z_j)^{-b}$
$q_P(y) =$	$+f_P$	$-f_Z$	$-\lambda_P\, y_P$	$-\kappa_P\, y_P^2$	$-\lambda'_P\, y_P$		
$q_{DOP}(y) =$		$+\sigma_{DOP}\, f_Z$	$+\lambda_P\, y_P$	$+\kappa_P\, y_P^2$	$+\lambda'_P\, y_P$	$-\lambda'_{DOP}\, y_{DOP}$	

Table B4. Equations for the NPZ-DOP model with $f_P = f_P(y_N, y_P, I_P)$, $f_Z = f_Z(y_P, y_Z)$ and $E_j = \bar{\sigma}_{DOP}(\bar{\sigma}_Z\, f_Z + \lambda_P\, y_P + \kappa_Z\, y_Z^2)\, dz_j$.

	Euphotic zone					All layers			Sinking
$q_N(y) =$	$-f_P$			$+\lambda_Z\, y_Z$				$+\lambda'_{DOP}\, y_{DOP}$	$+E_j\, \partial_z (z/z_j)^{-b}$
$q_P(y) =$	$+f_P$	$-f_Z$	$-\lambda_P\, y_P$			$-\lambda'_P\, y_P$			
$q_Z(y) =$		$+\sigma_Z\, f_Z$		$-\lambda_Z\, y_Z$	$-\kappa_Z\, y_Z^2$		$-\lambda'_Z\, y_Z$		
$q_{DOP}(y) =$		$+\sigma_{DOP}(\bar{\sigma}_Z\, f_Z$	$+\lambda_P\, y_P$		$+\kappa_Z\, y_Z^2)$	$+\lambda'_P\, y_P$	$+\lambda'_Z\, y_Z$	$-\lambda'_{DOP}\, y_{DOP}$	

Table B5. Equations for the NPZD-DOP model with $f_P = f_P(y_N, y_P, I_P)$ and $f_Z = f_Z(y_P, y_Z)$.

	Euphotic zone					All layers				Sinking
$q_N(y) =$	$-f_P$				$+\lambda_Z\, y_Z$			$+\lambda'_D\, y_D$	$+\lambda'_{DOP}\, y_{DOP}$	
$q_P(y) =$	$+f_P$	$-f_Z$	$-\lambda_P\, y_P$			$-\lambda'_P\, y_P$				
$q_Z(y) =$		$+\sigma_Z\, f_Z$		$-\kappa_Z\, y_Z^2$	$-\lambda_Z\, y_Z$		$-\lambda'_Z\, y_Z$			
$q_D(y) =$		$+\bar{\sigma}_{DOP}\,(\bar{\sigma}_Z\, f_Z$	$+\lambda_P\, y_P$	$+\kappa_Z\, y_Z^2)$				$-\lambda'_D\, y_D$		$+\partial_z w(z)\, y_D$
$q_{DOP}(y) =$		$+\sigma_{DOP}\,(\bar{\sigma}_Z\, f_Z$	$+\lambda_P\, y_P$	$+\kappa_Z\, y_Z^2)$		$+\lambda'_P\, y_P$	$+\lambda'_Z\, y_Z$		$-\lambda'_{DOP}\, y_{DOP}$	

Fasham, M. J. R., ed.: Ocean Biogeochemistry. The Role of the Ocean Carbon Cycle in Global Change., Global Change – The IGBP Series, Springer, Berlin et al., 2003.

Fennel, K., Losch, M., Schröter, J., and Wenzel, M.: Testing a marine ecosystem model: sensitivity analysis and parameter optimization, Journal of Marine Systems, 28, 45–63, doi:10.1016/S0924-7963(00)00083-X, 2001.

Griewank, A. and Walther, A.: Evaluating derivatives: principles and techniques of algorithmic differentiation, Society for Industrial and Applied Mathematics (SIAM), 2008.

Kelley, C. T.: Solving nonlinear equations with Newton's method, SIAM, Philadelphia, 2003.

Khatiwala, S.: A computational framework for simulation of biogeochemical tracers in the ocean, Global Biogeochemical Cycles, 21, doi:10.1029/2007GB002923, 2007.

Khatiwala, S.: Fast spin up of Ocean biogeochemical models using matrix-free Newton-Krylov, Ocean Modelling, 23, 121–129, doi:10.1016/j.ocemod.2008.05.002, 2008.

Khatiwala, S.: TMM framework web page, http://www.ldeo.columbia.edu/%7Espk/Research/TMM/ (last access: 14 June 2016), 2013.

Khatiwala, S., Visbeck, M., and Cane, M.: Accelerated simulation of passive tracers in ocean circulation models, Ocean Modelling, 9, 51–69, 2005.

Knoll, D. and Keyes, D.: Jacobian-free Newton–Krylov methods: a survey of approaches and applications, Journal of Computational Physics, 193, 357–397, 2004.

Kriest, I., Khatiwala, S., and Oschlies, A.: Towards an assessment of simple global marine biogeochemical models of different complexity, Progress In Oceanography, 86, 337–360, doi:10.1016/j.pocean.2010.05.002, 2010.

Marshall, J., Adcroft, A., Hill, C., Perelman, L., and Heisey, C.: A finite-volume, incompressible Navier Stokes model for studies of the ocean on parallel computers, Journal of Geophysical Research, 102, 5753–5766, 1997.

McKay, M. D., Beckman, R. J., and Conover, W. J.: Comparison of three methods for selecting values of input variables in the analysis of output from a computer code, Technometrics, 21, 239–245, 1979.

Merlis, T. M. and Khatiwala, S.: Fast dynamical spin-up of ocean general circulation models using Newton–Krylov methods, Ocean Modelling, 21, 97–105, 2008.

Paltridge, G. W. and Platt, C. M. R.: Radiative Processes in Meteorology and Climatology, Elsevier, New York, doi:10.1002/qj.49710343713, 1976.

Saad, Y.: Iterative Methods for Sparse Linear Systems, The Society for Industrial and Applied Mathematics (SIAM), 2003.

Saad, Y. and Schultz, M.: GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems, SIAM Journal on Scientific and Statistical Computing, 7, 856–869, doi:10.1137/0907058, 1986.

Sarmiento, J. L. and Gruber, N.: Ocean Biogeochemical Dynamics, Princeton University Press, Princeton et al., 2006.

Schartau, M. and Oschlies, A.: Simultaneous data-based optimization of a 1d-ecosystem model at three locations in the north Atlantic: Part I - method and parameter estimates, Journal of Marine Research 61, pp. 765–793, 2003.

Siewertsen, E., Piwonski, J., and Slawig, T.: Porting marine ecosystem model spin-up using transport matrices to GPUs, Geoscientific Model Development, 6, 17–28, doi:10.5194/gmd-6-17-2013, 2013.

Stoer, J. and Bulirsch, R.: Introduction to Numerical Analysis, Springer, New York, 3rd edn., 2002.

Walker, D. W. and Dongarra, J. J.: MPI: A Standard Message Passing Interface, Supercomputer, 12, 56–68, 1996.

Wang, D.: A note on using the accelerated convergence method in climate models, Tellus A, 53, 27–34, doi:10.1034/j.1600-0870.2001.01134.x, 2001.

Zumbusch, G. W.: Dynamic Load Balancing in a Lightweight Adaptive Parallel Multigrid PDE Solver., in: PPSC, SIAM, Philadelphia, 1999.

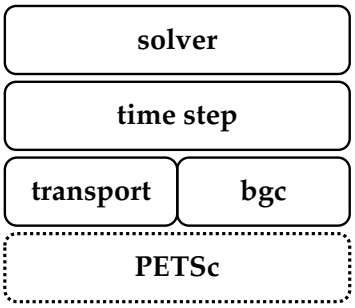

Figure 21. Implementation layers of the Metos3D simulation package (cf. Section 5.1).

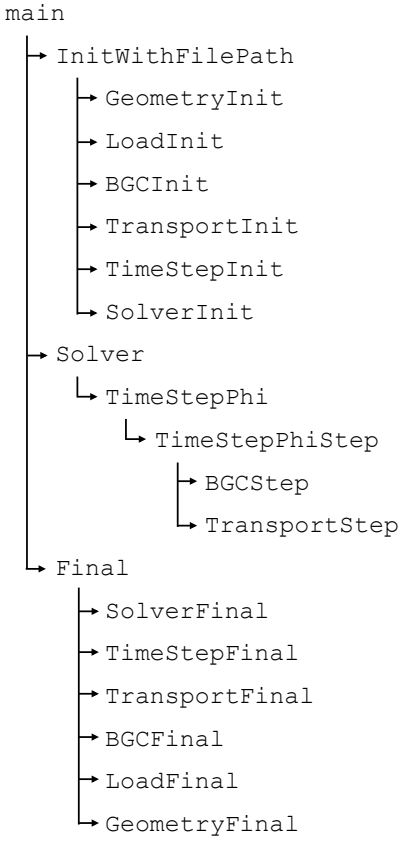

Figure 22. ~~Schematic of~~ Call graph for the ~~implementation structure~~ computation of ~~Metos3D~~a steady annual cycle(cf. Section 5.1).

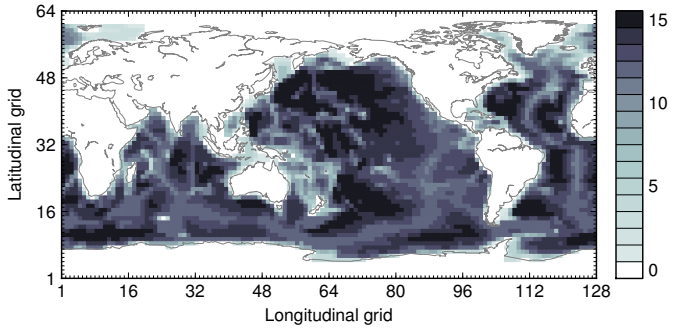

Figure 23. Land-sea mask (geometric data) of the used numerical model. Shown are the number of layers per grip point. Note that the Arctic has been filled in.

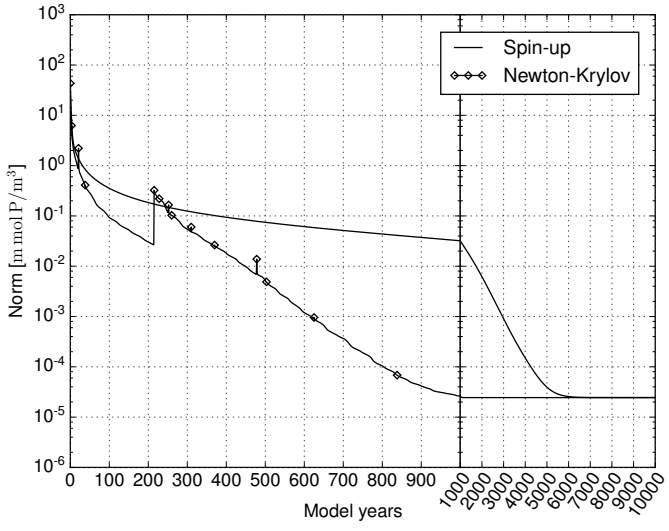

Figure 24. *MITgcm-PO4-DOP model:* Convergence towards an annual cycle. *Spin-up:* norm of difference between initial states of consecutive model years (solid line). *Newton-Krylov:* residual norm at a Newton step (diamond) and norm of the GMRES residual during solving (solid line in-between).

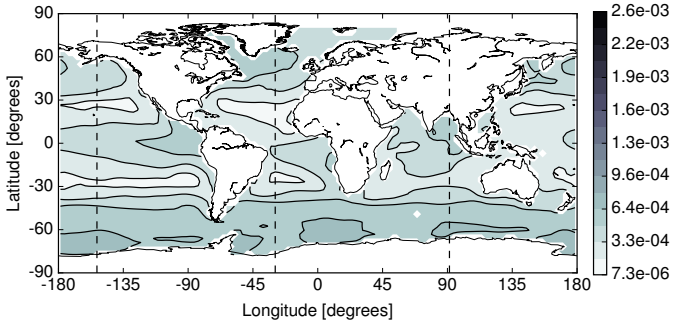

Figure 25. *MITgcm-PO4-DOP model:* Difference between the phosphate concentration of the spin-up and the Newton solution at the first layer (0 – 50 m) in the Euclidean norm. Units are $\mathrm{mmol\,P\,m^{-3}}$.

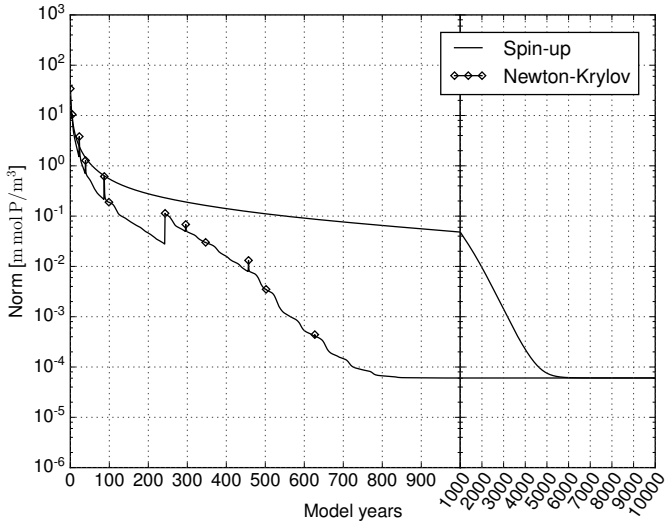

Figure 26. *N model:* Convergence towards an annual cycle using ~~a~~ spin-up and ~~a~~ Newton-Krylov solver.

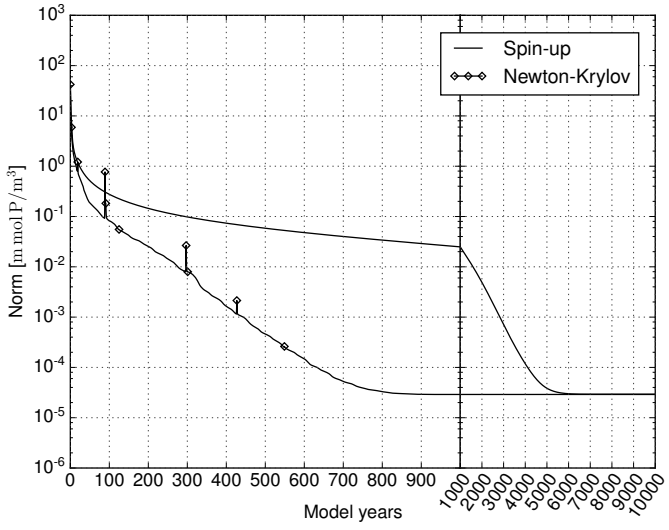

Figure 27. *N-DOP model:* Convergence towards an annual cycle using a spin-up and a Newton-Krylov solver.

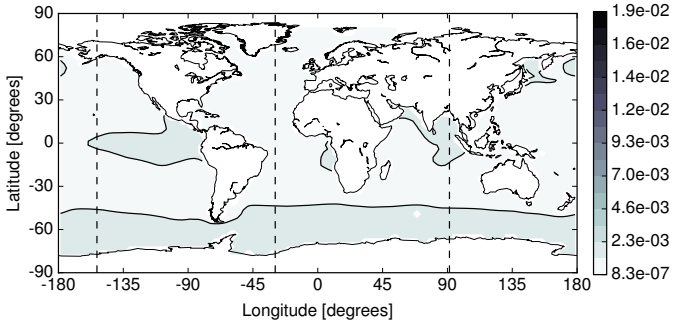

Figure 28. *N model:* Difference between the phosphate concentration of the spin-up and the Newton solution at the first layer (0 – 50 m) in the Euclidean norm. Units are $\mathrm{mmol\,P\,m^{-3}}$.

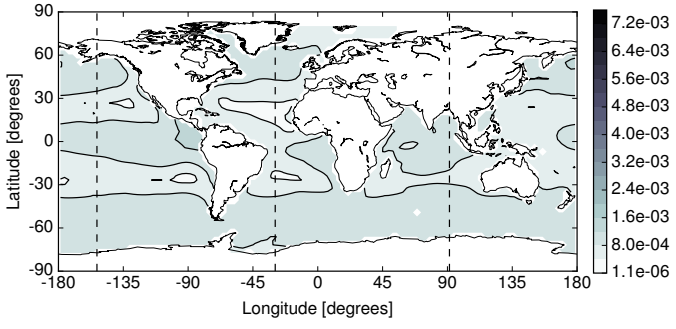

Figure 29. *N-DOP model:* Difference between the phosphate concentration of the spin-up and the Newton solution at the first layer (0 – 50 m) in the Euclidean norm. Units are $\mathrm{mmol\,P\,m^{-3}}$.

MITgcm-PO4-DOP model: Difference between the spin-up and Newton solution at the first layer (0 – 50 m) in the Euclidean norm.

N model: Difference between the spin-up and Newton solution at the first layer (0 – 50 m) in the Euclidean norm.

N-DOP model: Difference between the spin-up and Newton solution at the first layer (0 – 50 m) in the Euclidean norm.

NP-DOP model: Difference between the spin-up and Newton solution at the first layer (0 – 50 m) in the Euclidean norm.

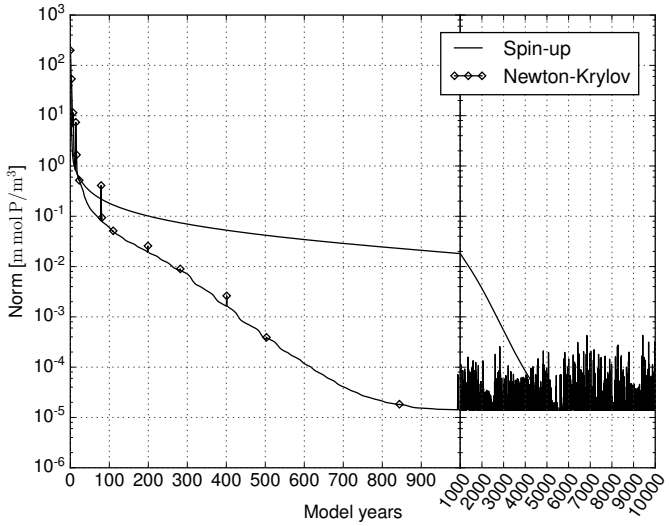

Figure 210. *NP-DOP model:* Convergence towards an annual cycle using a spin-up and a Newton-Krylov solver.

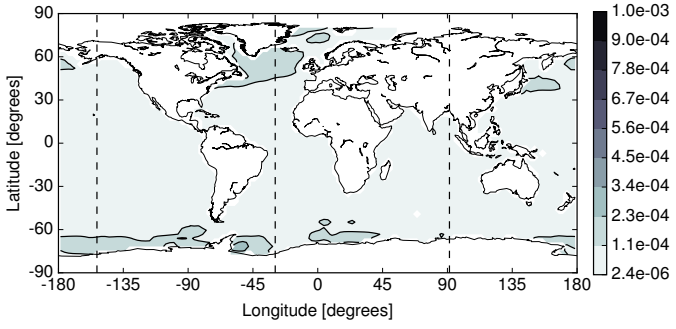

Figure 211. *NP-DOP model:* Difference between the phosphate concentration of the spin-up and the Newton solution at the first layer (0 – 50 m) in the Euclidean norm. Units are $\mathrm{mmol\,P\,m^{-3}}$.

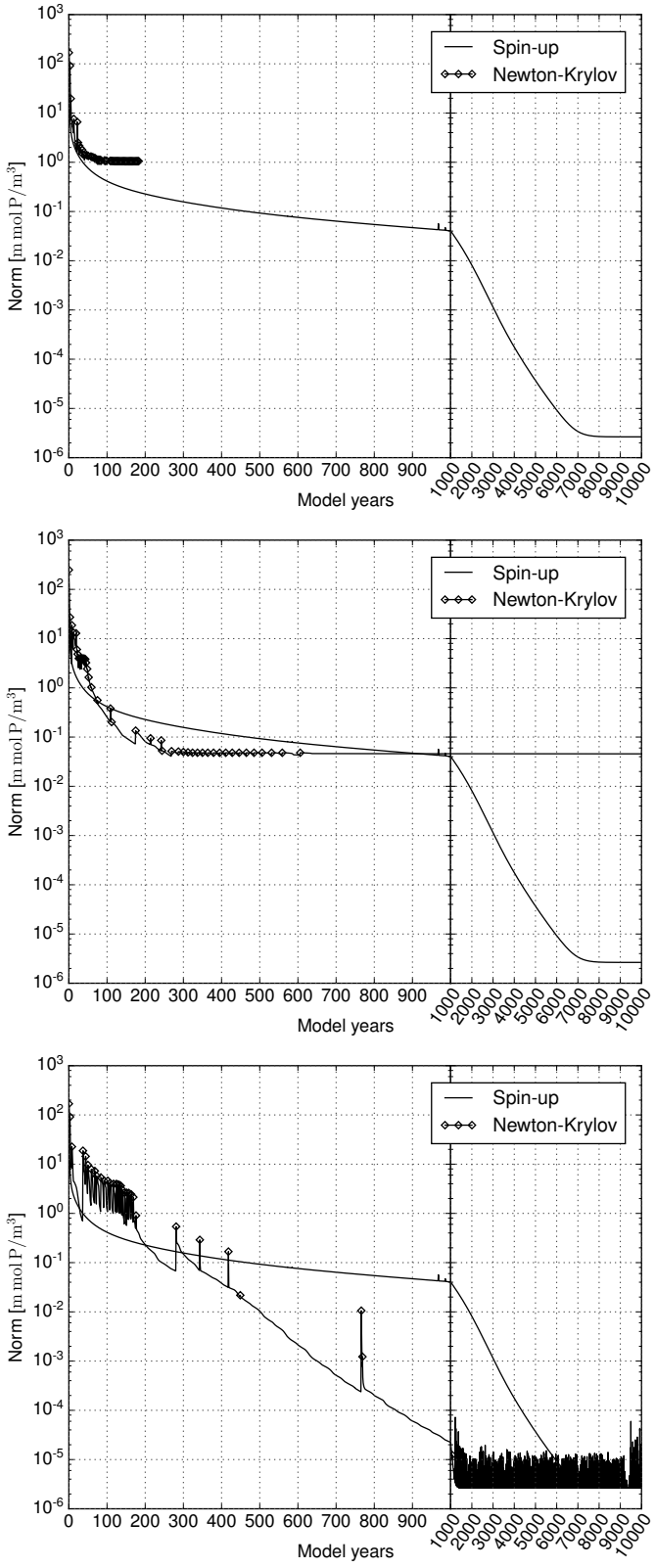

Figure 212. *NPZ-DOP model:* Convergence towards an annual cycle using a spin-up and a Newton-Krylov solver. *Top:* Default Newton-Krylov setting. *Middle:* ~~Changed initial~~ Initial value altered to ~~0.5425 m mol P m^{-3}~~ 0.5425 mmol P m^{-3} for all tracers. *Bottom:* ~~Changed inner~~ Inner accuracy altered to $\gamma = 0.3$.

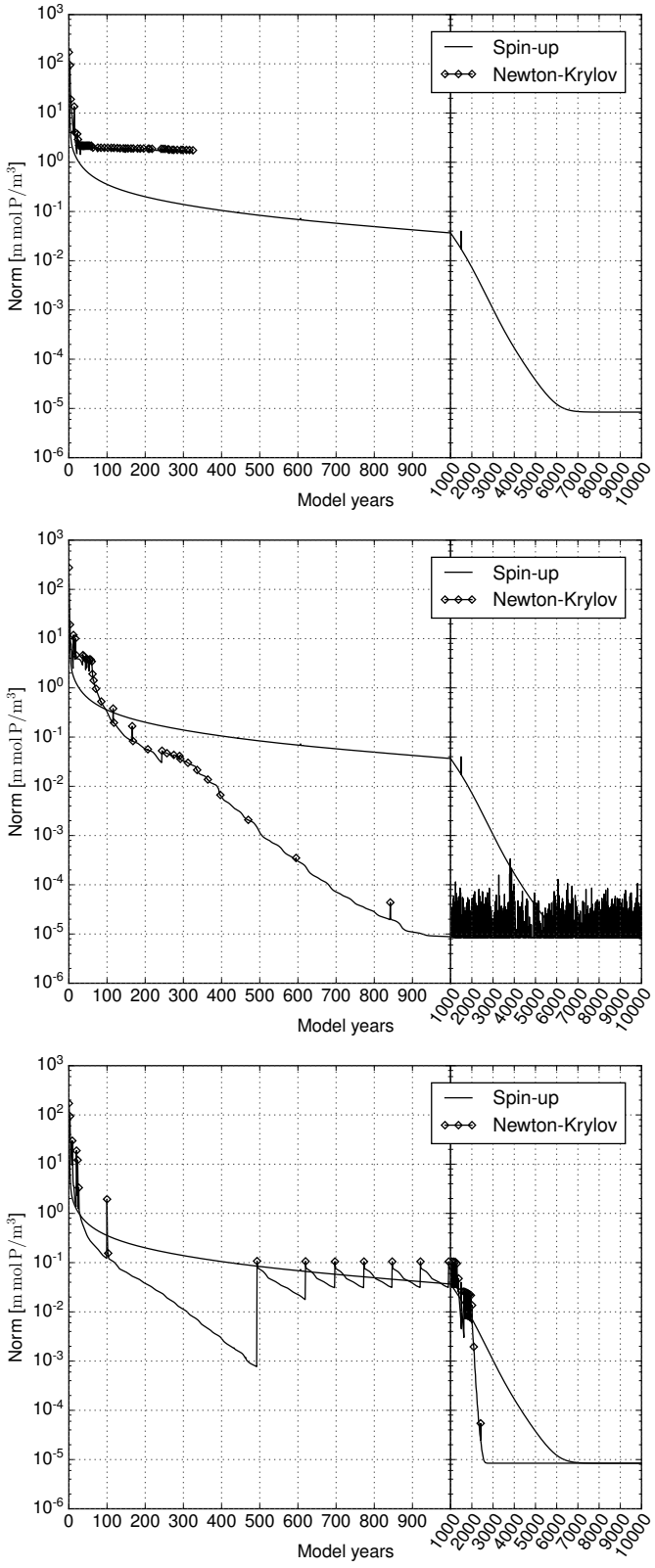

Figure 213. *NPZD-DOP model:* Convergence towards an annual cycle using a spin-up and a Newton-Krylov solver. *Top:* Default Newton-Krylov setting. *Middle:* ~~Changed initial~~ Initial value altered to ~~0.0434 m mol P m^{-3}~~ 0.0434 mmol P m^{-3} for all tracers. *Bottom:* ~~Changed inner~~ Inner accuracy altered to $\gamma = 0.3$.

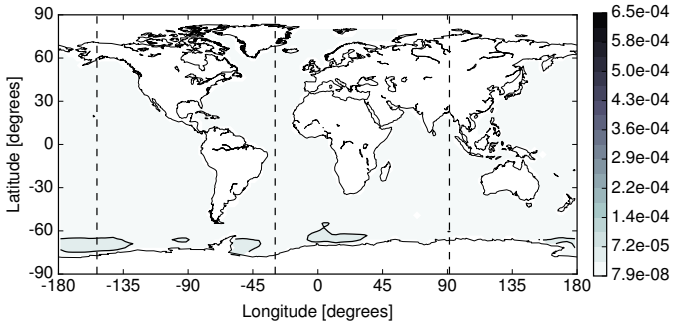

Figure 214. *NPZ-DOP model:* Difference between the phosphate concentration of the spin-up and the Newton solution at the first layer (0 – 50 m) in the Euclidean norm. Units are $\mathrm{mmol\,P\,m^{-3}}$.

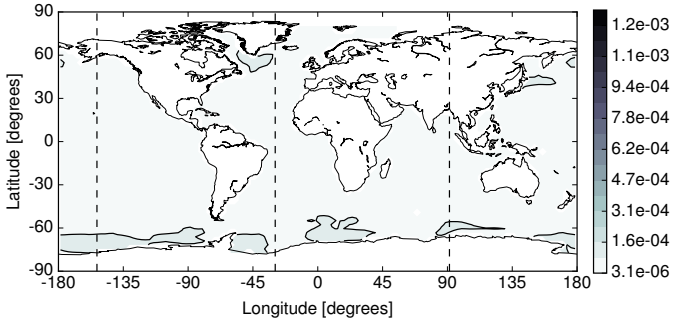

Figure 215. *NPZD-DOP model:* Difference between the phosphate concentration of the spin-up and the Newton solution at the first layer (0 – 50 m) in the Euclidean norm. Units are $\mathrm{mmol\,P\,m^{-3}}$.

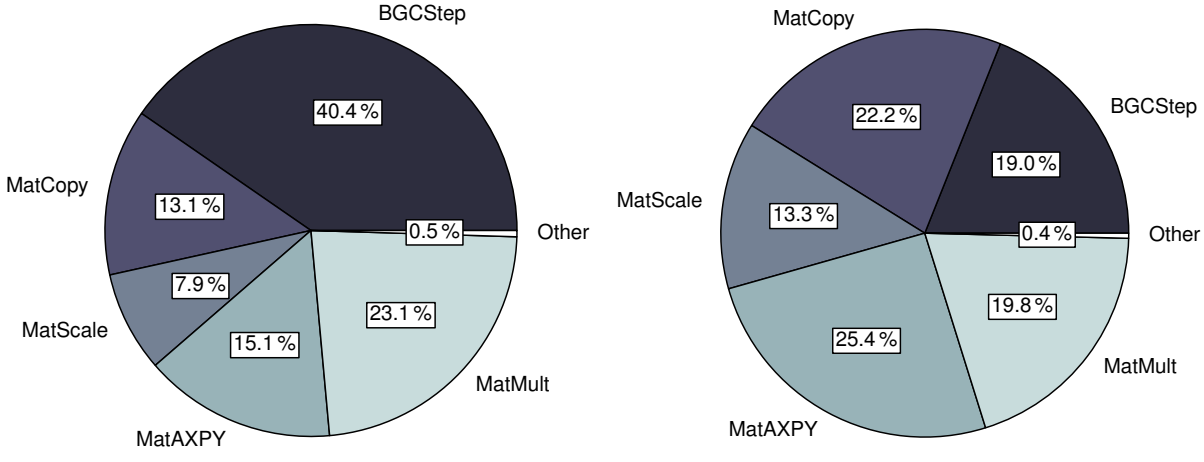

Figure 216. Distribution of ~~the~~ computational time among main operations during ~~the~~ integration of ~~a~~ one model year. *Left:* MITgcm-PO4-DOP model. *Right:* N model.

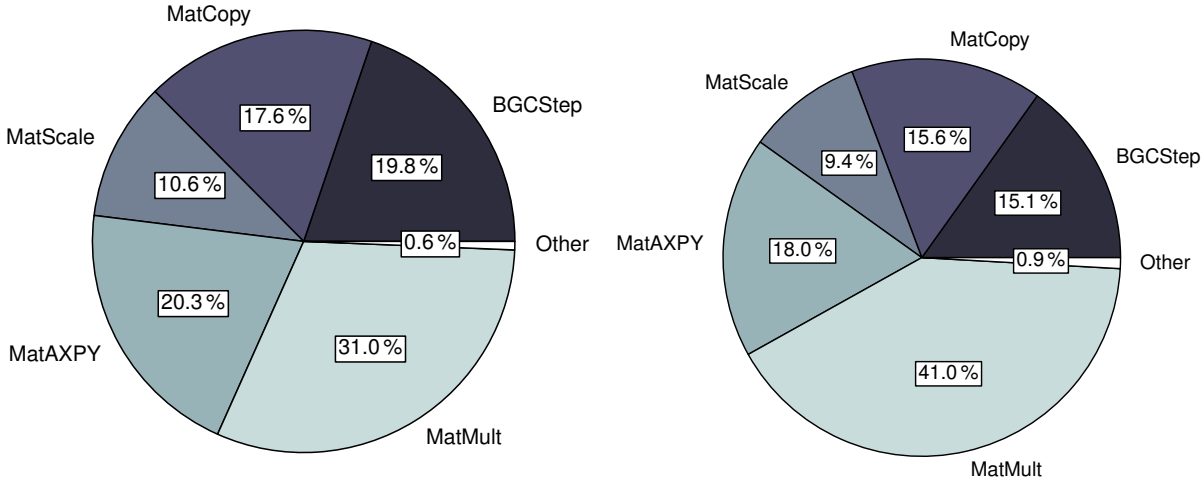

Figure 217. Distribution of ~~the~~ computational time among main operations during ~~the~~ integration of ~~a~~ one model year. *Left:* N-DOP model. *Right:* NP-DOP model.

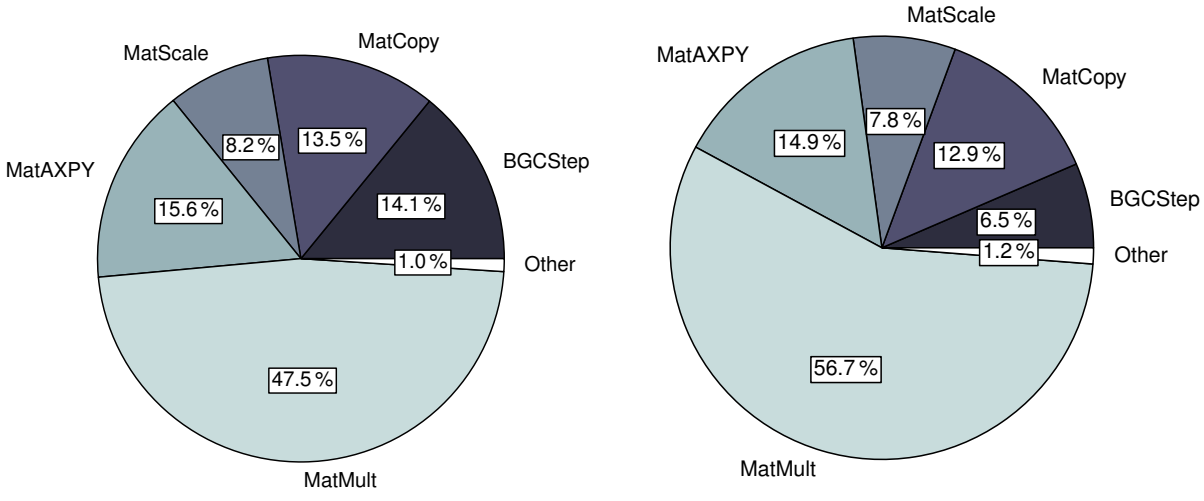

Figure 218. Distribution of ~~the~~ computational time among main operations during ~~the~~ integration of ~~a~~ one model year. *Left:* NPZ-DOP model. *Right:* NPZD-DOP model.

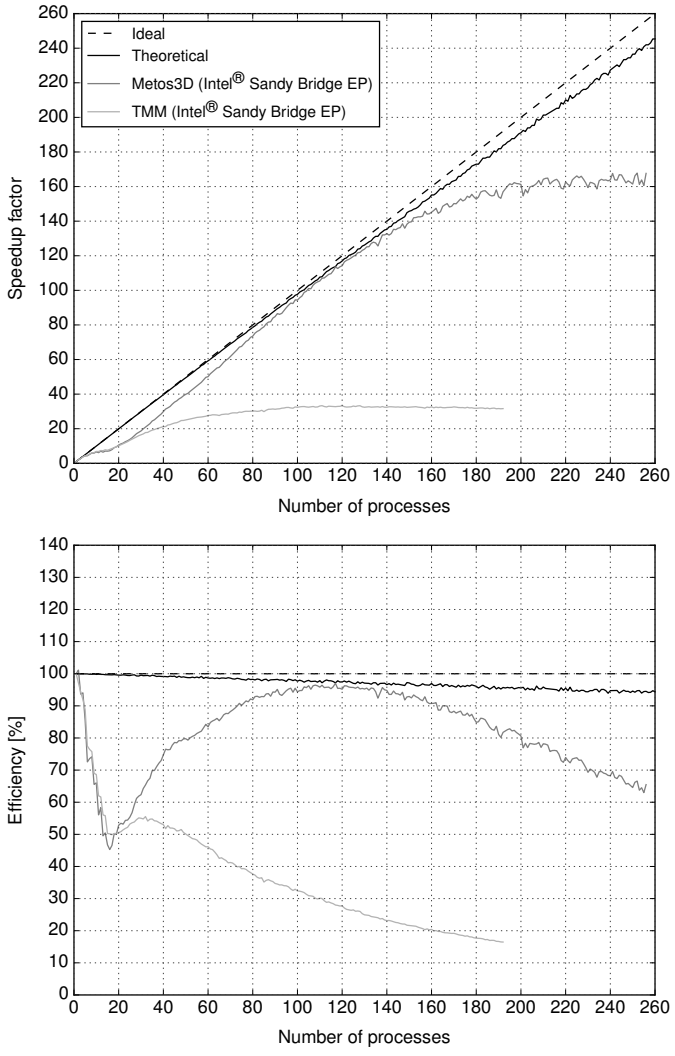

Figure 219. *MITgcm-PO4-DOP model:* Ideal and actual ~~speedup factor as well as~~ speed-up factors and efficiency of parallelized computations. ~~Here, the notion~~ The term 'theoretical' here refers to the ~~used~~ use of load distribution as introduced in Section 6.3~~, i.e. a simulation run on an idealized hardware.~~

Algorithm 1: ~~Load balancing~~ `Phi` (ϕ)

Input : initial condition: $(t_0, \boldsymbol{y}_0)$, time step: Δt, number of time steps: n_t, implicit matrices: $\mathbf{A}_{imp}$, explicit matrices: $\mathbf{A}_{exp}$, parameters: $\boldsymbol{u} \in \mathbb{R}^m$, boundary data: $\boldsymbol{b}$, domain data: $\boldsymbol{d}$

Output: final state: $\boldsymbol{y}_{out}$

~~$w = 0$~~ $\boldsymbol{y}_{in} = \boldsymbol{y}_0$;

~~$n_{p,1\ldots N} = 0$;~~ **for** $j = 1, \ldots, n_t$ **do**

  $t_j =$ mod $(t_0 + (j-1)\Delta t, 1.0)$;

  $\boldsymbol{y}_{out} =$ `PhiStep` $(t_j, \Delta t, \mathbf{A}_{imp}, \mathbf{A}_{exp}, \boldsymbol{y}_{in}, \boldsymbol{u}, \boldsymbol{b}, \boldsymbol{d})$;

  $\boldsymbol{y}_{in} = \boldsymbol{y}_{out}$;

end

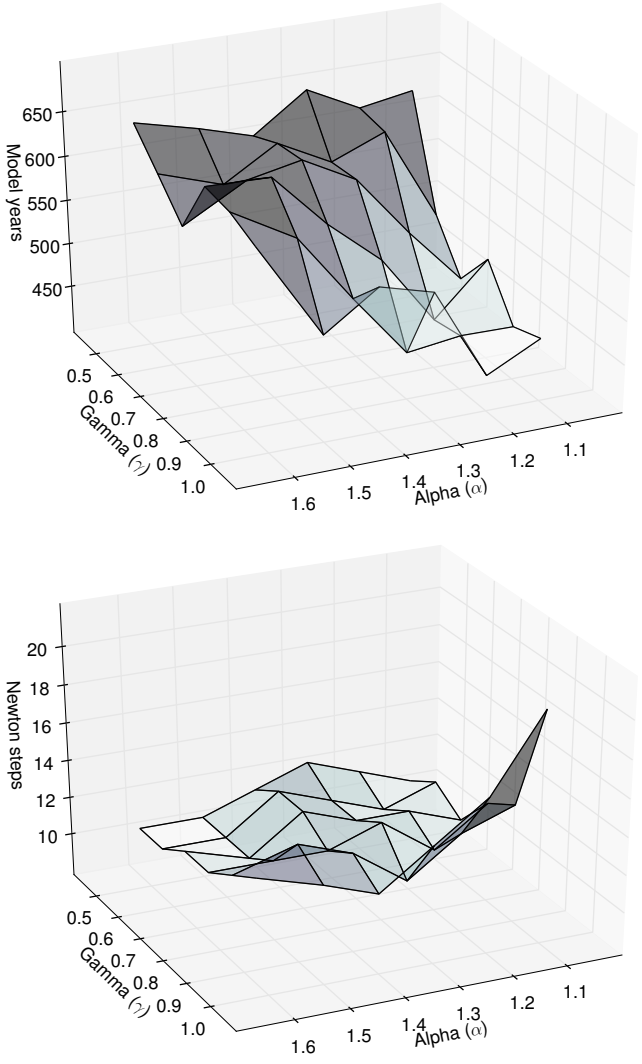

Figure 220. *MITgcm-PO4-DOP model:* Number of model years and Newton steps required for the computation of the annual cycle $\boldsymbol{y}(\boldsymbol{u}_d)$ as a function of different convergence control parameters α and γ (cf. Equation (8)).

Algorithm 2: ~~Interpolation~~ `PhiStep` (φ_j)

Input : point in time: t_j, time step: Δt, implicit matrices: $\mathbf{A}_{imp}$, explicit matrices: $\mathbf{A}_{exp}$, current state: $\boldsymbol{y}_{in}$, parameters: $\boldsymbol{u} \in \mathbb{R}^m$, boundary data: $\boldsymbol{b}$, domain data: $\boldsymbol{d}$

Output: next state: $\boldsymbol{y}_{out}$

1 ~~$w = t * n_{data} + 0.5$~~ $\boldsymbol{q} = \texttt{BGCStep}(t_j, \Delta t, \boldsymbol{y}_{in}, \boldsymbol{u}, \boldsymbol{b}, \boldsymbol{d})$;

2 ~~$\beta = \text{mod}(w, 1.0)$~~ $\boldsymbol{y}_w = \texttt{TransportStep}(t_j, \mathbf{A}_{exp}, \boldsymbol{y}_{in})$;

3 ~~$j_\beta = \text{mod}(\text{floor}(w), n_{data})$~~ $\boldsymbol{y}_w = \boldsymbol{y}_w + \boldsymbol{q}$;

4 ~~$\alpha = (1.0 - \beta)$~~ $\boldsymbol{y}_{out} = \texttt{TransportStep}(t_j, \mathbf{A}_{imp}, \boldsymbol{y}_w)$;

5 ~~$j_\alpha = \text{mod}(\text{floor}(w) + n_{data} - 1, n_{data})$;~~

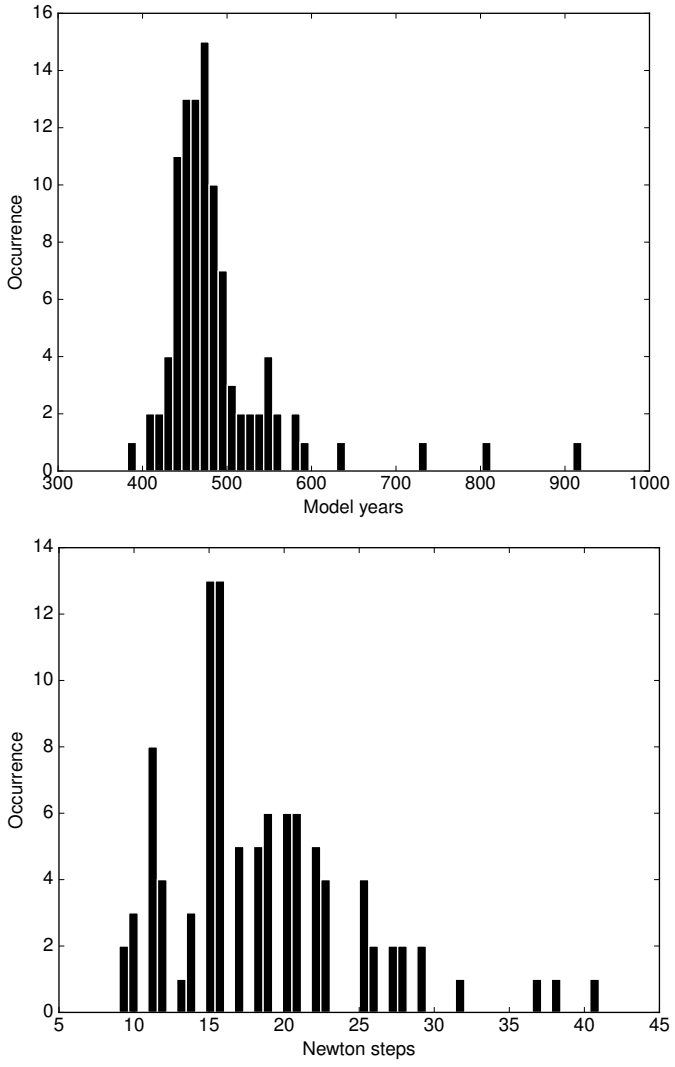

Figure 221. Distribution of number of model years and Newton steps required for the computation of ~~a~~ one annual cycle using 100 random parameter samples (cf. Section 6.5).

Algorithm 3: ~~Phi (ϕ)~~ Load balancing

Input : vector length: n_x, number of profiles: n_p, profile lengths: $(n_{x,k})_{k=1}^{n_p}$, number of processes: N
Output: profiles per process: $(n_{p,i})_{i=1}^{N}$

~~$y_{in} = y_0$~~ $w = 0$;
$n_{p,1,\dots,N} = 0$; **for** $k = 1, \dots, n_p$ **do**
    $i = \text{floor}(((w + 0.5 * n_{x,k})/n_y) * N)$;
    $n_{p,i} = n_{p,i} + 1$;
    $w = w + n_{x,k}$;
end

Table 26. Vertical layers of the numerical model, in meters.

Layer	Depth of layer bottom	Thickness of layer (Δz)
1	50	50
2	120	70
3	220	100
4	360	140
5	550	190
6	790	240
7	1080	290
8	1420	340
9	1810	390
10	2250	440
11	2740	490
12	3280	540
13	3870	590
14	4510	640
15	5200	690

Table 27. Parameters implemented in the MITgcm-PO4-DOP model. Specified are the location within the parameter vector, the symbol used by Dutkiewicz et al. (2005) and the value used for the computation of the reference solution ($\mathbf{u}_d$). Shown are furthermore the lower ($\mathbf{b}_l$) and upper ($\mathbf{b}_u$) boundaries used for the parameter samples experiment.

$\mathbf{u}$	Symbol	$\mathbf{u}_d$	$\mathbf{b}_l$	$\mathbf{b}_u$	Unit
u_1	κ_{remin}	0.5	0.25	0.75	y^{-1}
u_2	α	2.0	1.5	200.0	$\mathrm{mmol\,P\,m}^{-3}$
u_3	f_{DOP}	0.67	0.05	0.95	1
u_4	κ_{PO_4}	0.5	0.25	1.5	$\mathrm{mmol\,P\,m}^{-3}$
u_5	κ_I	30.0	10.0	50.0	$\mathrm{W\,m}^{-1}$
u_6	k	0.02	0.01	0.05	m^{-1}
u_7	a_{remin}	0.858	0.7	1.5	1

Algorithm 4: ~~`PhiStep` (φ)~~ Interpolation

Input : point in time: $t \in [0, 1[$, number of data points: n_{data}
Output: weights: α, β, indices: j_α, j_β

1 ~~$\mathbf{q} = \texttt{BGCStep}(t_j, \Delta t, \boldsymbol{y}_{in}, \boldsymbol{u}, \boldsymbol{b}, \boldsymbol{d})$~~ $w = t * n_{data} + 0.5$;
2 ~~$\boldsymbol{y}_w = \texttt{TransportStep}(t_j, \mathbf{A}_{exp}, \boldsymbol{y}_{in})$~~ $\beta = \mathrm{mod}(w, 1.0)$;
3 ~~$\boldsymbol{y}_w = \boldsymbol{y}_w + \mathbf{q}$~~ $j_\beta = \mathrm{mod}(\mathrm{floor}(w), n_{data})$;
4 ~~$\boldsymbol{y}_{out} = \texttt{TransportStep}(t_j, \mathbf{A}_{imp}, \boldsymbol{y}_w)$~~ $\alpha = (1.0 - \beta)$;
5 $j_\alpha = \mathrm{mod}(\mathrm{floor}(w) + n_{data} - 1, n_{data})$;

Listing 1. Fortran 95 implementation of the coupling interface for biogeochemical models.

```
subroutine metos3dbgc(ny, nx, nu, nb, nd, dt, q, t, y, u, b, d)
    integer :: ny, nx, nu, nb, nd
    real*8  :: dt, q(nx, ny), t, y(nx, ny), u(nu), b(nb), d(nx, nd)
end subroutine
```

Table 28. Parameter values used for the solver experiments with the N, N-DOP, NP-DOP, NPZ-DOP and NPZD-DOP model hierarchy.

Parameter	N	N-DOP	NP-DOP	NPZ-DOP	NPZD-DOP	Unit
k_w	0.02	0.02	0.02	0.02	0.02	m^{-1}
k_c			0.48	0.48	0.48	$(\mathrm{mmol\,P\,m}^{-3})^{-1}\mathrm{m}^{-1}$
μ_P	2.0	2.0	2.0	2.0	2.0	d^{-1}
μ_Z			2.0	2.0	2.0	d^{-1}
K_N	0.5	0.5	0.5	0.5	0.5	$\mathrm{mmol\,P\,m}^{-3}$
K_P			0.088	0.088	0.088	$\mathrm{mmol\,P\,m}^{-3}$
K_I	30.0	30.0	30.0	30.0	30.0	$\mathrm{W\,m}^{-2}$
σ_Z				0.75	0.75	1
σ_{DOP}		0.67	0.67	0.67	0.67	1
λ_P			0.04	0.04	0.04	d^{-1}
κ_P			4.0			$(\mathrm{mmol\,P\,m}^{-3})^{-1}\mathrm{d}^{-1}$
λ_Z				0.03	0.03	d^{-1}
κ_Z				3.2	3.2	$(\mathrm{mmol\,P\,m}^{-3})^{-1}\mathrm{d}^{-1}$
λ'_P			0.01	0.01	0.01	d^{-1}
λ'_Z				0.01	0.01	d^{-1}
λ'_D					0.05	d^{-1}
λ'_{DOP}		0.5	0.5	0.5	0.5	y^{-1}
b	0.858	0.858	0.858	0.858		1
a_D					0.058	d^{-1}
b_D					0.0	$\mathrm{m\,d}^{-1}$

Table 29. Difference in the Euclidean ($\|\cdot\|_2$) and volume-weighted ($\|\cdot\|_{2,V}$, cf. Eq. (4)) norms between the spin-up ($\boldsymbol{y}_S$) and the Newton ($\boldsymbol{y}_N$) solution for all models. The total volume of the ocean used here is $V \approx 1.174 \times 10^{18}\,\mathrm{m}^3$. Solutions for models NPZ-DOP and NPZD-DOP were produced by experiments with altered inner accuracy or initial value, respectively.

Model	$\|\boldsymbol{y}_S - \boldsymbol{y}_N\|_2$	$\|\boldsymbol{y}_S - \boldsymbol{y}_N\|_{2,V}$
MITgcm-PO4-DOP	1.460e-01	7.473e+05
N	4.640e-01	2.756e+06
N-DOP	2.421e-01	1.199e+06
NP-DOP	7.013e-02	3.633e+05
NPZ-DOP	1.421e-02	8.514e+04
NPZD-DOP	3.750e-02	2.062e+05

Table 210. Minimum, maximum, average and standard deviation of computational time for one model year as well as the computing time per tracer is shown. All computations were performed on a single core Intel Xeon® E5-2670 CPU at 2.6 GHz.

	Min	Max	Avg	StdDev	Min per tracer
N	112.53 s	112.87 s	112.79 s	0.09	112.53 s
N-DOP	142.96 s	143.30 s	143.12 s	0.11	71.48 s
NP-DOP	160.32 s	161.28 s	160.86 s	0.30	53.44 s
NPZ-DOP	185.46 s	185.70 s	185.53 s	0.07	46.37 s
NPZD-DOP	193.99 s	194.63 s	194.09 s	0.19	38.80 s