Dear Andrew,
Dear referees,

as suggested we changed the focus of the publication and concentrated purely on simulation, which better suits the title.
We skipped all parts that refer to optimization (Section 7.7, including WOA data, twinexperiments
etc) in the updated version.
We added remarks that the intended purpose of the „simulation package“ is a later optimitazion (to motivate the „O“ in the abbreviation METOS3D for the software).
To emphasize the focus „simulation“, we additionally implemented and included results for six biogeochemical models in total now.

Please find our detailed answers to the referees’ letters below:

Anonymous Referee #1
Received and published: 13 July 2015
#
General comments

In this paper the authors present a newly assembled toolkit (Metos3D) for the imple-
mentation of two solvers based on PETSc library. I understand Metos3D is meant to
generalize the coupling of transport matrices and source/sink models with the possibil-
ity to use a Newton solver as well as a fixed point iteration (spin-up). This should
save the user the effort to develop a coding interface every time a new source/sink
model is introduced.
However, my understanding outlined above (if at all correct) comes from a lot of guess-
work. The authors do not explain clearly what the objective of the work is. There is a
paragraph that was meant for this (page 4404, lines 18-26) but it should be improved.
I think the Introduction before that paragraph did a fair job in introducing the problem,
outlining the three components put together here to tackle it but then from line 10 of
page 4404 it gets a bit confused and the first sentence of the objective paragraph
(Lines 18-19) sounds oddly out of place to me.

We have re-written the mentioned paragraph of the introduction and emphasized the objectives of our work.

To improve the introduction we added the following paragraph before
(page 4404, line 10, a reference in parenthesis refers to the
gmdd-8-4401-2015.pdf discussion paper), where it got confusing.
See: metos3d-simpack-jpits-diff.pdf, lines 73-89.
'''
Optimization methods usually require hundreds of model evaluations.
As a consequence, an environment for optimization of marine ecosystems
that is intended by (and mentioned in the name of) our software Metos3D
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 intended usage.
As a consequence, we impose a high standard of flexibility w.r.t.
interchange of models and solvers.
'''

Then we repeated the components of metos3d again.
See: metos3d-simpack-jpits-diff.pdf, lines 95-100.
'''
In this work we combine three of them in our software,
namely the so-called off-line simulation,
the option for the use of Newton's method for the computation of steady annual
cycles (as an alternative to a spin-up)
and spatial parallelization with high scalability.
'''

We refined the paragraph about steady annual cycles and the Newton solver.
See: metos3d-simpack-jpits-diff.pdf, lines 112-126.
'''
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 the mapping that integrates the model variables over one year model time.
In this sense, a spin-up is a fixed-point iteration.
By a straighforward procedure, this fixed-point problem can be equivalently
transformed into the problem of finding the root(s) of a nonlinear mapping.
For this kind of problem,
Newton-type methods \citep[cf.][Chapter 6]{DenSch96} are
well known for their superlinear convergence.
'''

Then we emphasized the importance of a geometry-adapted load balancing algorithm.
See: metos3d-simpack-jpits-diff.pdf, lines 132-144.
'''

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.
Parallel software that
employs transport matrices and
targets a multi-core distributed-memory architecture
requires appropriate data types and linear algebra operations.
Additionally, 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.
'''

Next, we emphasized the objectives of our work
as was intended on (page 4404, lines 18-26).
See: metos3d-simpack-jpits-diff.pdf, lines 151-165.
'''
The objective of this work is to unite the mentioned three
performance-enhancing techniques (off-line computation via
transport matrices, Newton method, and highly scalable parallelization)
in a software environment with rigorous modularity and
complete open-source accessibility.
Here, modularity refers to the separation of data pre-processing and
simulation and the flexibility of coupling any water column-based
biogeochemical model with minimized implementation effort.
For this purpose,
we defined a model interface that permits any number of tracers,
parameters as well as boundary and domain data.
Its flexibility we show by using both an available biogeochemical
model \citep[][]{DuSoScSt05},
taken from the MITgcm ocean model, as well as a suite of more complex ones,
which is included in our software package.
'''

In general, I think the paper requires a better structural organization in order to
improve its readability. Furthermore, it needs to highlight better what is the novelty
here and why should a user use this toolkit and what for. What is the contribution with
respect to the work of Khatiwala (Ocean Modelling 23 (2008) 121–129) where a matrix-free
Newton– Krylov solver was applied to a similar framework?

This is now mentioned in the introduction (in the last but one paragraph) as well as in the
Section 7.

They are four main differences to the TMM software provided by Samar Khatiwala:
1. Open source: TMM uses Matlab — we purely rely on open source software
2. Modularity: TMM combines the extraction of matrices and their application, for the former
Matlab is needed — we decoupled both, provide software for the latter
3. Newton solver convergence: TMM was only used (or results were published) for one model —
we applied it for six, we studied and compared the convergence, moreover studied model and
solver parameters' effect on performance, we discussed solver tuning options in the case of
poor convergence
4. Improved load balancing: see speed-up figure, our software scales much better than TMM on
parallel machines

Subsequent, we made clear what is the novelty of our approach compared to Samars TMM.
See: metos3d-simpack-jpits-diff.pdf, lines 168-187
'''
As a result, the work of \citet[][]{Kha08}
could be extended by numerically showing convergence for all six
abovementioned models without applying preconditioning.
Moreover, a detailed profiling analysis for the simulation with
the different biogeochemical models shows 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,
and in this respect superiority over other approaches,
including the one used in \citet[][]{TMMweb13}.
'''

In its present form I'm afraid the manuscript wouldn't be able to encourage a model user
to download and get acquainted with Metos3D.
#
Specific comments

Sections 2 and 3 could be shortened or maybe moved to supplement material or to an
appendix. Section 4 seems to get to the core of the novel contribution of this work and
I think it could be merged with Section 6. This merged section would benefit from
schematics or a flowcharts to help the reader to better understand how Metos3D works.
The description of the implementation of the toolkit in layers (section 6.1) is an
example of something that would probably be better explained in a schematic.
In Section 5 it would help to give 5.3.2 the same title as 5.2 (aren't they the same

thing?).

From here, in our opinion,
the 'latexdiff'-document explains the differences best.

Sectons 2 and 3 were not put in appendix.
We use them to introduce the notation and explain the discretization
as well as the resulting off-line transport by matrices.
See comments below on comments of Referee #2.

To the achieve 'a better structural organization'
old sections (page 4410, line 1 to page 4413, line 23), namely
5 Periodic solutions
5.1 Spin-up
5.2 Newton
5.3 Convergence
5.3.1 Spin-up
5.3.2 Newton
were removed.
See: metos3d-simpack-jpits-diff.pdf, lines 606-725.

They were replaced by
4 Steady annual cycles
4.1 Computation by spin-up
4.2 Computation by inexact Newton method
See: metos3d-simpack-jpits-diff.pdf, lines 358-537.

The next sections must be read in the new context anyway.

7.2 Solver, rewritten (due to new models).
See: metos3d-simpack-jpits-diff.pdf, lines 1028-1090.

8 Conclusions, rewritten (due to new focus).
See: metos3d-simpack-jpits-diff.pdf, lines 1298-1495.

Appendix, new (description of model hierarchy).
See: metos3d-simpack-jpits-diff.pdf, lines 1521-1598.

Former Sections 4 and 5 have been interchanged.
Old Section 4 (new Section 5) and Section 6 are now followed by each other.
Sections 2, 3, 5 old (= 4 new) have been shortened and reformulated.
We added a schematic figure in Section 6.

Section 6.2: I know PETSc has its own load balancing algorithm. How is your
procedure different?

Our procedure is a load balancing algorithm for vector parts (i.e. vertical profiles) with
different length.
We are not aware of such an algorithm in PETSc (3.3).

In Section 7 results are presented. In Section 7.1.1 (Model) at
lines 2-3 of page 4420, it is said that a model wrapper of the original source code is
implemented. Is this something that the user of Metos3D will have to do for every
source/sink model?

If the model conforms to the biogeochemical interface, no.

Is some sort of guidance or template provided?

We think that the description of the interface and the (now newly included) model suite will
be helpful.

Figure 3 compares the initial state of the converged annual cycle with WOA. I don’t see
the point of this comparison. Your solution should be compared with the solution coming
from the spinup of the MITgcm coupled with the same biogeochemical model, if available,
of course. I don’t believe the skill of the model in reproducing real-world observations
is the point here. It does not say anything about Metos3D.

We omitted all comparison to data now. We compare the results of the spin-up with the results
of the Newton solver now.

Technical corrections
----
page 4403, line 12: ...is often TOO high, even at LOW resolutions,...

Corrected.

page 4410, line 2: “With those two building blocks”. It does not hurt to remind which
two blocks you mean.

Sections have been revised basically.

Page 4411, line 5: k=1,....?

Sections have been revised basically.

Page 4414, line 5: “The latter includes (?)”

Corrected.

Page 4414, line 23: “The next TWO layers”

Corrected.

Page 4415, lines 6-7: this last sentence is confusing. Consider reformulating it.

We did.

Page 4418, line 1-2: “We compare the simulation results with others” others what?

Obsolete. We compare solver results now.

Page 4424, lines 11-12: consider rephrasing this one.

Section has been revised basically.

Page 4426, lines 5-6: consider explaining why it is so interesting.

We considered this, but without any further experiments we don't want to speculate.

Page 4428, lines 2: what was the “intended purpose”? Remind it here.

Obsolete.

———————————————————————————————
Anonymous Referee #2
Received and published: 15 July 2015
#
This manuscript presents a steady state offline solver for marine biogeochemical mod-
els using two alternative approaches: a iterative procedure towards the fixed point
solution or a Newtonian equation solver of the residual norm. While the subject of the
work is generally relevant to the journal presenting a novel tool with a high potential
for application in the scientific community, it falls short on a couple of important
points that need addressing if it was to be considered for a full GMD publication.
My main concern is the lack of clarity on what the purpose of the tool in its current
state is and what it actually delivers.
The abstract promotes it as a tool for parameter
identification and a lot of the intrudctory and final discussion mention optimisation.
However, it is not stated what is intended by parameter identification and how the tool
would achieve it. I assume from the discussion that the intention of the authors is the
indentification of an optimal parameter set for a given biogeochemical model, while what
the tool actually delivers is a periodic steady state solver for biogeochemical models
using offline ocean physics.

This was misleading in the first version. We now omitted the optimization as mentioned earlier.

This can without doubt be a valuable element for a parameter
optimisation toolkit, but is nevertheless only one element of it and moreover it doesn’t
address the core of the actual optimisation problem, e.g. what should a model be op-
timised against. In addition, it doesn’t give a benchmark that would allow compariso
against other optimistation tools. (The work does give some permformance indications for
the steady state solver, but no generalised performance indications for the optimi-
sation process.) Moreover, the authors demonstrate themselves in their example that
the application within the optimisation process is still premature. On this background,
I would suggest to change the pitch of the manuscript towards what the tool actually is
done for (at least to my understanding), and what it actually delivers successfully and
reliably, i.e. the periodic steady state solution of the biogeochemical model. I see no
reason to limit the tool to a specific application in optimisation that is then treated
only superficially and insufficiently if this is given as the main purpose of the tool.
On the contrary I can see a series of other valuable applications to any form of large
ensemble experiment and examples may be given in the discussions to highlight the utility
of such a tool beyond optimisation.

In this respect, we cleaned up the manuscript basically.

The new abstract states now clearly that we present
a comprehensive high-performace toolkit for the computation
of steady annual cycles with a general programming interface
for water column models.
See: metos3d-simpack-jpits-diff.pdf, lines 1-51.

As a second point the manuscript lacks generally in
clarity (some examples below) and requires a considerable review in grammar and style.
For future submissions, I would strongly suggests the authors to review their manuscripts
before initial submission on these terms (maybe with the help of a native speaker) as I
believe a lot of the points given below could have been addressed in this process lead-
ing to a much more beneficial review. Reviewing the work in its current form required a
considerable amount of assumptions of what was actually intended.

We improve the text.

Some comments in detail:
----
pg 4402 line 2: what is intended with parameter indentification?
pg 4403 line 8: when talking about biogeochemical models and their validation in generic
term, the obvious question concerning the estimation of an optimal model parametrisation
forehand, is what the model should be optimised against? I believe this will be highly
application dependent.

This becomes obsolete since we changed the focus of the paper.

pg 4403 line 23: "acceptable loss of accuracy" involved in the
splitting of ocean physics and biogeochemical processes: any references?

Khatiwala et al. 2005

pg 4404 line 4-10: I'd suggest to move this to the later section where residual and norm
are introduced, it becomes much clearer then, particularly to modellers with a less
numerical background.
pg 4406, line 15: the dimensionless time "1" here refers to one intra-annual time step,
while in the above section (lines 4,6) it refers to one periodic step, i.e. one year.
These should be distinct by or using different variables for time within the annual cycle
and in the iteration procedure, or by explicitly using time units.
pg 4406, line 24 onwards: I'd suggest to introduce necessity for the split explicit and
implicit treatment of physical processes first and then specify it's application to the
offline solver in order to facilitate understanding for readers that are unfamiliar with
the problem.

In this regard, Sections 2 - 4 have been revised basically.

pg 4407 eq 2, lines 16,17: difference between A and A' should be clarified.

Different notation was used:
A' became L now, but the whole section was shortended and clarified.

pg 4407 line 22 - pg4408, line 1: I'd expect the sufficient resolution of the tracer
transport process on monthly time steps to be highly configuration and application
dependent, rather than hold generically.

We agree, but did not want to elaborate on this here. Anyway, the text is formulated more
generally now.

pg 4408, lines 11-15: "Generally, we assume
that a tracer model is implemented for a single water column, synonymously called profile
in the following. This assumption does not constrain the interface for the future and,
it actually simplifies the current software implementation." The interface to the
biogeochemical models is the main point of the tool and being clear here is essential to
encourage potential users. I'd suggest spending a couple of words here stating assumptions
and limitations clearer, i.e. - any "client" model must be able to take-up its states from
the interface in water column format. - no geometrical information on horizontal vicinity
of the vertical profiles is preserved in the interface. - any model that requires
horizontal structure in it's internal computation requires modification in the internals
of the tool. I realise that the vast majority of biogeochemical models currently used will
fullfill these requirements, but they should be explicit.

The remarks have been incorporated.
pg 4409 eq 3, where have the indices y,k gone?
This seems to be a missunderstanding. n_{y,k} as the length of a profile has been fixed.
We tried to make this clear.

pg 4409, last paragraph, what's the purpose
of the initialisation and finalisation routines.

Added to Section 6.1: '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.'

pg 4410 lines 3,4: confusion in the use of 1 in the time dimension, see above

This sections have been revised basically.

pg 4412 line 16: Why is the unweighted norm used?

It is equivalent to the weighted norm. We clarified this in the present manuscript.

pg 4414 line 13 "repository of the simulation package"

Corrected.

pg 4414 line 23 "The next both layers" -> next two layers

Corrected.

pg 4416 lines 12-15 "Thus, the matrices and vectors are linearly interpolated to the cur-
rent time step during the iteration. The files of a specific data set are interpreted as
averages of the time intervals they represent. Consequently, we interpolate in between the
associated centers of these intervals." If linear interpolation is used the result will be
non-conservative, which should be noted.

At the end of the last paragraph of section 3 the sources of errors of the transport matrix approach are summarized.

pg 4416 line 25: how are the weights alpha and beta determined, i.e. is this a linear
interpolation?

Yes, see Section 3.

pg. 4419 lines 19-23: so the effective state variables are two, all others are
diagnostics? Should be made clearer.

Yes, that is right.

pg. 4419, line25,26: what is the "introduced convention for directory structure"?

Made explicit now.

pg. 4420 line 20: You may want to consider hosting the binary data outside the git
repository.

In this regard, we considered alot. However, the data will stay at github.
But, we will use GitHub Large File Support in the future.

pg. 4421 line 27: again, wouldn’t this number be application and configuration dependent?

Yes, whole section was revised.

pg. 4423 line 27 ratio of what?

Section was revised.

pg. 4426 line 12 state the origin of the reference solution and its purpose
pg. 4427 lines 15-21 are unclear to me. Maybe the figure would
help, but unfortunately the labels are unreadable at this scale.

Section has been removed.

pg 4428 line 2 "intended purpose", what is the intended purpose?

Obsolete.

pg 4428 line 22 "was somehow "natural"" what’s meant by this?

Reference omitted.

pg 4428 line 28 "computationally still too complex", I suppose the authors intend
too expensive?

Yes.

pg 4429 line 3-6 I fail to see why a suitable choice of the time step
would have complicated the verification.

For a different time step new matrices must be *prepared*.
This processes must be explained.
As it is part of the matrix preparation process
We decided this should not be part of this manussript.

pg 4429 lines 9-13 Here the authors clearly state that the solver tool in its current
form fails to deliver the intended purpose, i.e. parameter identification, see general
comments above.

Reference omitted.
pg 4429 lines 20-21 what’s the expected flexibility?

pg 4429 lines 24-25 not a sentence

Corrected.

Figures 3,4,6,7,8,9,10 are unreadable and require larger labels.

Figures 3 (surface), 6 (speedup), 7 (convergence control), 8 (samples): labels have been enlarged. Figures 4 (slices), 9 (twin) and 10 (twin) have been removed.

Manuscript prepared for Geosci. Model Dev.
with version 2014/09/16 7.15 Copernicus papers of the LaTeX class copernicus.cls.
Date: 29 February 2016

Metos3D: A Marine Ecosystem Toolkit for Optimization and Simulation in 3-D – 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. ~~A general programming interface for parameter identification for~~ We designed and implemented a modular software framework for the off-line simulation of steady cycles of 3-D marine ecosystem models ~~is introduced. A comprehensive solver software for periodic steady-states is implemented that includes a fixed point iteration (spin-up) and a Newton solver. The software is~~ based on the ~~Portable, Extensible Toolkit for Scientific Computation (PETSc) library and uses transport matrices for efficient off-line simulation in 3-D~~transport matrix approach. ~~In addition to the usage of PETSc's parallel data structures and PETSc's Newton solver, an own load balancing algorithm is implemented.~~

~~A simple verification is carried out using a well investigated biogeochemical model for phosphate (PO_4) and dissolved organic phosphorous (DOP) with 7 parameters~~It is intended to be used in parameter optimization and model assessment experiments. ~~The model is coupled via the interface to transport matrices that correspond to a longitudinal and latitudinal resolution of 2.8125° and 15 vertical layers~~We defined a software interface for the coupling of a general class of water column-based biogeochemical models, with six of them being part of the package. ~~Initial tests show that both solvers and the load balancing algorithm work correctly~~The framework offers both spin-up/fixed-point iteration and Jacobian-free Newton method for the computation of steady states. ~~Further experiments demonstrate the robustness of the Newton solver with respect to parameter variations~~The Newton method converged with standard setting for four models, and with a change in one solver parameter or the initial guess for two more complex ones. ~~Moreover,~~ For all considered models, both methods delivered the same steady state (within a reasonable precision) on convergence, with the ~~numerical tests reveal that, with optimal control settings, the Newton solver converges at least 6 times fastertowards a solution than the spin-up.~~

~~However, additional twin experiments reveal differences between both solvers regarding a derivative-based black-box optimization~~Newton iteration being in general 6 times faster. ~~Whereas an optimization run with spin-up-based model evaluations is capable to identify model parameters of a reference solution, Newton-based model evaluations result in an inaccurate gradient approximation~~For one exemplary model, we investigated the effect of both the biogeochemical and the Newton solver parameters on the performance. We performed a profiling analysis for all considered models, in which the number of tracers had a dominant impact on the overall performance. We implemented a geometry-adapted load balancing procedure which showed nearly optimal scalability up to a high number of parallel processors.

1 Introduction

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

A wide range of marine ecosystem models needs to be validated, i.e. assessed regarding their ability to re-

produce ~~the real world system~~real world data. This involves a professional discussion of simulation results and, ~~preferably~~moreover, 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 hundreds of model evaluations. As a consequence, an environment for optimization of marine ecosystems that is intended by (and mentioned in the name of) our software Metos3D 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. As a consequence, we impose a high standard of flexibility w.r.t. interchange of models and solvers.

The computational effort of a fully coupled simulation, i.e. a simultaneous and interdependent computation of ocean circulation and tracer transport in three spatial dimensions, ~~however, is often to~~ is very high, even at ~~lower resolution, considering optimization methods that may require hundreds of model evaluations~~low resolution. Moreover, the complexity increases additionally if annual cycles are investigated, in which one model evaluation involves a long time integration (the so-called spin-up) until an equilibrium state under given forcing is reached (cf. Bernsen et al., 2008).

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

Off-line simulation offers a fundamentally reduced computational cost compared to an acceptable loss of accuracy. The principle idea is to pre-compute transport data for passive tracers. Such an approach has been adopted by Khatiwala et al. (2005) to introduce the so-called Transport Matrix Method (TMM; Khatiwala, 2013). The authors make use of matrices to store results from a general circulation model and to apply them later on to arbitrary 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).

From the mathematical point of view, ~~an~~ a steady annual cycle is ~~obtained by solving a time dependent, periodic system of nonlinear~~ a periodic solution of a system of (in this case) nonlinear parabolic partial differential equations. ~~The~~ This periodic solution is a ~~sequence of states and its initial is a fixed point of a mapping that is used to integrate given variables over a model year~~ fixed-point of the mapping that integrates the model variables over one year model time. ~~This fixed point is a zero of an equivalent nonlinear residual as well (cf. Kelley, 2003)~~In this sense, a spin-up is a fixed-point iteration. By a straighforward procedure, this fixed-point problem can be equivalently transformed into the problem of finding the root(s) of a nonlinear mapping. ~~In that case~~For this kind of problem, Newton-type methods (cf. Dennis and Schnabel, 1996, Chapter 6) are well known for their superlinear convergence~~towards a solution~~. In combination with a Krylov subspace approach, a Jacobian-free scheme can be realized that is based only on evaluations of one model year (cf. Knoll and Keyes, 2004; Merlis and Khatiwala, 2008; Bernsen et al., 2008).

~~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).

The ~~main objective of our work , though, is to stay focused on a general coupling for biogeochemical models and its embedment into an optimization context~~objective of this work is to unite the mentioned three performance-enhancing techniques (off-line computation via transport matrices, Newton method, and highly scalable parallelization) in a software environment with rigorous modularity and complete open-source accessibility. Here, modularity refers to the separation of data pre-processing and simulation and the flexibility of coupling any water column-based biogeochemical model with minimized implementation effort. ~~Thus, we define a general programming~~ For this purpose, we defined a model interface that permits any number of tracers, parameters as well as boundary and domain data. ~~We implement a comprehensive, transport matrix based solver software around the method call and map its arguments onto a flexible option system of the final executable~~Its flexibility we show by using both an available biogeochemical model (Dutkiewicz et al., 2005), taken from the MITgcm ocean model, as well as a suite of more complex ones, which is included in our software package. Our software allows for choosing among spin-up/fixed-point iteration and Newton method, where for the latter tuning options are studied. As a result, the work of Khatiwala (2008) could be extended by numerically showing convergence for all six abovementioned models without applying preconditioning. Moreover, ~~for purposes~~

~~of usability we provide an install script for the toolkit and all the material we used to perform the presented numerical experiments~~a detailed profiling analysis for the simulation with the different biogeochemical models shows how the number of tracers impacts the overall performance. Finally, an adapted load balancing method is presented. ~~This includes data preparation, result parsing and visualization scripts~~It shows nearly optimal scalability up to 128 processes, and in this respect superiority over other approaches, including the one used in Khatiwala (2013).

The ~~remainder of this~~ paper is organized as follows. In Sections ~~2–4~~ 2 and 3 we describe the marine ecosystem dynamics ~~, shortly~~ and recapitulate the transport matrix approach~~and define the biogeochemical model interface~~. In Sections ~~5–7 we discuss~~ 4 we summarize the two options for the computation of steady cycles/periodic solutions, ~~go into details of the implementation~~ namely the fixed-point and Newton iteration, where for the latter we also discuss tuning options to achieve better convergence. In Sections 5 and ~~present~~ 6, we describe design and implementation of our software package, and Section 7 shows ist applicability and performance in several numerical results. ~~Finally,~~ In Section 8 ~~concludes our work and~~ we draw conclusions and in Section 9 ~~describes~~ describe how to obtain the source code.

In the Appendix, we summarize the model equations and parameter settings of the model suite we used for this work and that is available together with the simulation software.

2 Marine ecosystem dynamics

We consider the following ~~off-line~~ tracer transport model, which is ~~described~~ defined by a system of ~~nonlinear parabolic differential equations defined on the unit time interval $I = [0,1[\subset \mathbb{R}$,~~ 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 $\Omega \subset \mathbb{R}^3$ ~~and its~~ with boundary $\Gamma = \partial\Omega$. ~~Throughout this work, the time interval is associated with one model year. For n tracers the system generally reads~~

$$\frac{\partial y_i}{\partial t} = \nabla \cdot (\kappa \nabla y_i) - \nabla \cdot (v\, y_i) + q_i(y,\boldsymbol{u},b,d),$$

~~where y_i is a tracer concentrations with $y_i : I \times \Omega \to \mathbb{R}$ and $y = (y_i)_{i=1}^n$ is a~~ Here $y_i : I \times \Omega \to \mathbb{R}$ denotes one single tracer concentration and $y = (y_i)_{i=1}^{n_y}$ the vector of all tracers. ~~Here, we neglect the~~ Since we are interested in long-time behavior and steady annual cycles, we assume that the time variable is scaled in years. We omit the additional dependency on the time and space coordinates ~~(t,x)~~ $(t,\boldsymbol{x})$ in the notation for brevity.

The transport of tracers in marine waters is ~~depicted by a diffusion and an advection term~~determined by diffusion and advection which is reflected in the first two linear terms on the right-hand side of (2). ~~The diffusion~~ Diffusion mixing coefficient $\kappa : I \times \Omega \to \mathbb{R}$ and ~~the~~ advection velocity field $v : I \times \Omega \to \mathbb{R}^3$ ~~are~~ may be regarded as given ~~(cf. Section 3).~~ data or have to be simulated together with (2) by an ocean model. ~~Note that both operators effect each tracer separately.~~ 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 the ecosystem are represented by the last term on the right-hand side of (2), i.e.

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

~~In contrast, a single component of~~ Often, the functions q_i are nonlinear and depend on several tracers, which couples the system. We will refer to the set of functions $q = (q_i)_{i=1}^{n_y}$ as "the biogeochemical model~~q_i may generally depend on all tracers~~". This model typically depends also on parameters. In the software we present in this paper these are assumed to be constant w. r. t. space and time, i.e.

$$q_i(y,\boldsymbol{u},b,d) = q_i(y_1,\dots,y_n,\boldsymbol{u},b,d).$$

we have $u = \boldsymbol{u} \in \mathbb{R}^{n_u}$. ~~Here, $b = (b_i)_{i=1}^{n_b}$ with $b_i : I \times \Gamma_s \to \mathbb{R}$ is a vector of boundary forcing data like~~ In the general setting of (2) this is not necessary. Boundary forcing (e.g. insolation or wind speed, ~~which is~~ defined on the ocean surface $\Gamma_s \subset \Gamma$. ~~Additionally, $d = (d_i)_{i=1}^{n_d}$ with $d_i : I \times \Omega \to \mathbb{R}$ is a vector of domain forcing data like~~) and domain forcing functions (e.g. salinity or temperature of the ocean water~~(cf. Section 5)~~) my also enter the biogeochemical model. ~~As mentioned in the introduction, the modelalso includes parameters that are optionally subject to optimization (cf. Table 19 as an example). They are summarized in the vector $\boldsymbol{u} \in \mathbb{R}^m$ and kept temporally as well as spatially constant during the computation of a model year~~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.

~~Additionally, homogeneous Neumann boundary conditions on the entire Γ~~ A reasonable setting are homogeneous Neumann conditions for all tracers y_i ~~are imposed~~on the entire boundary Γ. ~~An initial condition (t_0,y_0) with $t_0 \in [0,1[$ and $y_0 = (y_i(t_0,x))_{i=1}^n$ is provided .~~ Moreover, a function $y_0(\boldsymbol{x}) = (y_i(0,\boldsymbol{x}))_{i=1}^{n_y}, \boldsymbol{x} \in \Omega$, has to be provided to solve an initial-boundary-value problem for (2).

3 Transport matrix approach

The transport matrix method (Khatiwala et al., 2005) is a method that allows fast simulation of tracer transport

assuming that the forcing data diffusion κ and advection velocity v are given. ~~Overall, we assume the given forcing data κ, v, b and d is periodic, i. e. $\kappa(t+1,x) = \kappa(t,x)$ for example~~The method is based on the discretized counterpart of (2). 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. ~~Accordingly, we solve the model equations by computing an annual cycle, which is~~ At time instant t_j, we denote by

- $\boldsymbol{y}_{ji} = (\boldsymbol{y}_i(t_j, \boldsymbol{x}_k))_{k=1}^{n_x}$ the vector of the values of the i-th tracer at all grid points,
- $\boldsymbol{y}_j = (\boldsymbol{y}_{ji})_{i=1}^{n_y} \in \mathbb{R}^{n_y n_x}$ a vector of ~~tracer concentrations with $y(t+1,x) = y(t,x)$ (cf. Section 4).~~ 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 the biogeochemical terms in the j-th time step.

4 ~~Transport matrices~~

For the boundary data only corresponding grid points are incorporated.

The ~~idea of transport matrices is based on the fact that diffusion and advection are linear mappings at every point in time. Hence, the model equations can be written as~~ transport matrix method approximates the discretized counterpart of (2) by

$$\frac{\partial y_i}{\partial t}(t) = L(t)\, y_i(t) + q_i(t, y(t), \boldsymbol{u}, b(t), d(t)),$$

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

~~where $L(t)$ comprises both and represents a timedependent linear operator~~The linear operators $\mathbf{L}_{exp,j}, \mathbf{L}_{imp,j}$ represent the parts of the transport term in (2) that are discretized explicitly and implicitly w. r. t. time, respectively. ~~Formally, its fully discrete equivalent is a sequence of matrices $(\mathbf{L}_j)_{j=1}^{n_t}$ with $\mathbf{L}_j = \mathbf{L}(t_j)$~~Consequently, these operators depend on the given transport data κ, v and thus on time. ~~Here, n_t is the number of time steps and $t_j = t_0 + (j-1)\Delta t$ denotes a specific point in time with $\Delta t = 1/n_t$~~The biogeochemical term is treated explicitly in (5) by an Euler step.

Since the transport effects each tracer separately 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. ~~Note that throughout this work an equidistant time step will be used.~~

~~However, the matrices that we use here represent the effect of an entire time step~~In Khatiwala et al. (2005), it is described how these matrices can be computed by running one step of an ocean model for an appropriately chosen set of basis functions for a tracer distribution. As a consequence, the partition of the transport operator in (2) into the explicit and implicit matrix depends on the operator splitting scheme used in the ocean model. Usually diffusion (or a part of it) is discretized implicitly, in our case vertical diffusion only. ~~They are extracted from a sophisticated general circulation model that implements a combination of an operator splitting scheme and an implicit and explicit time step approach (cf. Temam, 1979)~~By this procedure, a set of matrix pairs $(\mathbf{A}_{exp,j}, \mathbf{A}_{imp,j})_{j=0}^{n_t-1}$ is obtained, which usually are sparse. ~~This requires code knowledge and implies a technical effort that is described by Khatiwala et al. (2005) for instance.~~ To reduce storing effort and to make the method feasible at all, only a smaller number of (in our case monthly) averaged matrices is stored. ~~As a general rule, once the discretization parameters are chosen,~~ From these, 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 sparse matrix-vector multiplications and evaluations of the biogeochemical model. Specifically, the ~~arrangement of the~~ implicit part of the time integration is now pre-computed and contained in $\mathbf{A}_{impl,j}$, which is the benefit of the method. The interpolation of the transport matrices, the ~~boundary and domain data and the tracer vectors are determined for further usage~~linearization of eventually used nonlinear discretization schemes (e.g. flux limiters), and disregarding the influence of the biogeochemistry back onto the circulation fields determine the approximation error of the method compared to a direct coupled computation.

~~The splitting scheme is reflected by the corresponding implicit and explicit matrices, respectively~~

4 Steady annual cycles

The purpose of the software presented in this paper is the fast computation of steady annual cycles of the considered marine ecosystem model. ~~Formally, an implicit transport matrix can be understood as the solution of the implicit time step and an explicit transport matrix as the application of the explicit time step, i. e.~~ A steady annual cycle is defined as periodic solution of (2) with period length 1 (year), thus satisfying

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

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

For the application of the transport matrix method, we assume that a set of matrices for one model year (generated with such kind of periodic forcing) is available, and that these are interpolated to 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

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

Assuming that the discrete model is completely deterministic, it suffices to satisfy this equation just for one j. Here, we compare solutions of the respective 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 (5) over one year. Here we omitted all other arguments except of $\boldsymbol{y}$ in the notation. Thus, a steady annual cycle is a fixed-point of the nonlinear mapping ϕ.

Since condition (10) will never be satisfied exactly in a simulation, we measure the periodicity using norms on $\mathbb{R}^{n_y n_x}$ for the residual of (10). We use the weighted Euclidean norm

$$\mathbf{A}_{imp,j} = (\mathbf{I} - \Delta t \mathbf{L}_{imp,j})^{-1}$$
$$\mathbf{A}_{exp,j} = (\mathbf{I} + \Delta t \mathbf{L}_{exp,j}).$$

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

~~Here,~~ for $\boldsymbol{z} \in \mathbb{R}^{n_y n_x}$ 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 to the continuous problem (2) is the ~~transport is split as $\mathbf{L}_j = \mathbf{L}_{imp,j} + \mathbf{L}_{exp,j}$ and $\mathbf{I}$ represents the identity.~~ discretized counterpart of the $\left(L^2(\Omega)\right)^{n_y}$-norm, where w_k is set to the volume of the k-th grid box. This norm we denote by $\|\boldsymbol{z}\|_{2,\Omega}$. Orther settings of the weights are possible. All these norms are equivalent with

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

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

Repeatedly applying iteration step (10) or – in other words – integrating in time with fixed forcing until convergence is reached, is termed spin-up. ~~Throughout this work, both matrix types are sparse~~It is well known by Banach's fixed-point theorem (cf. Stoer and Bulirsch, 2002) 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 unique fixed-point for all initial values $\boldsymbol{y}^0$. ~~The implicit matrix $\mathbf{L}_{imp,j}$ comprises vertical diffusion only ,~~ This result still holds on weaker assumptions (cf. Ciric, 1974). The 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 $\boldsymbol{q}_j'(\boldsymbol{y}_j)$ of the nonlinear biogeochemical model at the current iterates. Typically, thousands of iteration steps (i.e. ~~a process within a water column that is computed and inverted independently of its vicinity.~~ model years) are needed in order to reach a steady cycle (cf. Bernsen et al., 2008). ~~The explicit matrix $\mathbf{L}_{exp,j}$ represents a (local) differential operator, which naturally has a sparse discrete representation.~~ The method offers only restricted options for convergence tuning, the only straightforward one being the choice of a different time steps Δt_j. To to so, the transport matrices have to be re-scaled accordingly. The natural 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.

~~Overall, the fully discrete iteration scheme for n tracers results in a block diagonal system~~

4.2 Computation by inexact Newton method

By defining $F(\boldsymbol{y}) := \boldsymbol{y} - \phi(\boldsymbol{y})$, the fixed-point problem (10) can be equivalently transformed into the problem of finding a root of $F : \mathbb{R}^{n_y n_x} \to \mathbb{R}^{n_y n_x}$. ~~The integration of state variables over a model yearconsists of sparse matrix vector multiplications and evaluations of the biogeochemical model.~~ This problem can be solved by Newton's method (cf. Dennis and Schnabel, 1996; **?**; 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 quadratic convergence. ~~For a fixed time index j it reads~~ Starting from an initial guess $\boldsymbol{y}^0$, in every step the linear system

$$\mathbf{y}_{j+1} = \mathbf{A}_{imp,j}'(\mathbf{A}_{exp,j}' \mathbf{y}_j + \Delta t \mathbf{q}_j(\mathbf{y}_j, \mathbf{u}, \mathbf{b}_j, \mathbf{d}_j))$$
$$= \varphi_j(\mathbf{y}_j, \mathbf{u}, \mathbf{b}_j, \mathbf{d}_j),$$

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

~~where $\mathbf{y}_j = (\mathbf{y}_i(t_j))_{i=1}^n$ combines all discrete tracer vectors~~has to be solved, followed by an update $\boldsymbol{y}^{m+1} = \boldsymbol{y}^m + \varrho \boldsymbol{s}^m$. ~~Accordingly, $\mathbf{A}'_{imp,j}$ and $\mathbf{A}'_{exp,j}$ denote block diagonal matrices with $\mathbf{A}_{imp,j}$ and $\mathbf{A}_{exp,j}$ as their identical blocks, respectively~~Here $\varrho > 0$ is a step-size that is chosen iteratively such that a sufficient reduction in $\|F(\boldsymbol{y}^m + \varrho \boldsymbol{s}^m)\|_2$ is achieved (cf. Dennis and Schnabel, 1996, Section 6.3).

The Jacobian $F'(\boldsymbol{y}^m)$ of F at the current iterate includes the derivative of one model year, thus it is not as sparse as the transport matrices themselves. As a consequence, a matrix-free version of Newton's method is applied: The linear system (10) 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 definite, we use the generalized minimal residual method (GMRES, Saad and Schultz, 1986). The ~~components of the tracer model are depicted by $\mathbf{q}_j$ with~~

$$\mathbf{q}_j(\mathbf{y}_j, \boldsymbol{u}, \mathbf{b}_j, \mathbf{d}_j) \quad = (\mathbf{q}_i(t_j, \mathbf{y}_j, \mathbf{u}, \mathbf{b}_j, \mathbf{d}_j))_{i=1}^n,$$

needed matrix-vector products can be interpreted as directional derivatives of F at the point $\boldsymbol{y}^m$ in direction $\boldsymbol{s}$. They can be approximated by a forward finite difference:

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

~~where the discrete boundary respectively domain data is represented by $\mathbf{b}_j = (\mathbf{b}_i(t_j))_{i=1}^{n_b}$ and $\mathbf{d}_j = (\mathbf{d}_i(t_j))_{i=1}^{n_d}$.~~ 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 would be an exact evaluation of the derivative using the forward mode of algorithmic differentiation (cf. Griewank and Walther, 2008).

~~Actually, only 12 implicit and 12 explicit matrices are extracted and stored, when the TMM data is prepared~~The above approximation of the Jacobian or directional derivative is one reason for this method to be called an *inexact* one. ~~They represent monthly averaged ocean circulation, but provide a sufficient accuracy at minimal storage requirements as shown by Khatiwala et al. (2005).~~The second reason is that the inner linear solver has to be stopped and thus is also not exact. Here we use a convergence control procedure based on the technique described by Eisenstat and Walker (1996). They stop 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

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

This approach avoids so-called over-solving, i.e. wasting inner steps when the current Newton step was not very successful. The ~~same applies for the given forcing~~latter is typically the case in the beginning of a Newton iteration. The ~~matrices as well as the boundary and domain data are interpolated later on to the current time step during the computation of a model year (cf. Section 6).~~ parameters γ and α can be used to influence this behavior in a linear and nonlinear way, respectively. Moreover, they are a subtle way to tune the solver. In contrast to a fixed-point iteration, Newton's method also in its damped version may only converge with an appropriately chosen initial guess $\boldsymbol{y}^0$. In a high-dimensional problem as our application (in $\mathbb{R}^{n_y n_x}$), it is a non-trivial task to find such initial guess if the method with the standard one (e.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 only a few inner iterations, it makes sense to increase this number by either decreasing γ or increasing α. We will give examples later on where exactly this strategy enables convergence at all.

Concerning the total effort of the inexact Newton solver and in order to compare its efficiency with the spin-up, we first note that one evaluation of F basically corresponds to one application of ϕ, i.e. one model year. Thus, each Newton step requires one evaluation of F as right-hand side in (10). Within the inner linear solver iteration, the initial guess is always taken as $\boldsymbol{s} = 0$. Thus, no computation is required for the first step. Each following inner iteration require sone additional evaluation of F to compute the second term in the numerator of the right-hand side of (6). Additionally, the line search may require additional eavluations of F. In total, the overall number of inner iterations plus the overall number of evaluations in the line search determine the number of necessary evaluations of F that can be compared to the necessary model years in the spin-up.

5 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. The aim is to link any model implementation with any number of tracers, parameters as well as boundary and domain data to the driver software. The coupling must additionally fit into an optimization context, and it must be compatible with Algorithmic Differentiation techniques (cf. Section 8).

Generally, we assume that a tracer model is implemented for a single water column, synonymously called profile in the following. This means no geometrical information on horizontal vicinity of the vertical profiles is preserved in the interface. Moreover, any client model must be able to take up its states from such profiles. 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.

However, this assumption does not constrain the interface for the future~~and, it actually simplifies the current software implementation~~. ~~Moreover, it reflects the fact that the~~ In fact, the most important non-local biogeochemical processes happen within a water column (cf. Evans and Garçon, 1997).

~~Thus~~Consequently, throughout this work, each discrete tracer vector is a collection of profiles. It can be understood as a sparse representation of a land-sea cuboid including only wet grid boxes. The geometry information is provided as a 2-D land-sea mask with additional designation of the number of vertical layers (cf. Figure 12). Hence, a vector length n_y is a sum of non-equidistant profiles, i.e.

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

where n_p is the number of profiles and ~~$(n_{y,k})_{k=1}^{n_p}$~~ $(n_{x,k})_{k=1}^{n_p}$ is a set of profile depths.

The evaluation of the whole ~~n~~ n_y tracer model for a fixed time index j consist then of separate model evaluations for each profile. For a fixed profile index k ~~with a depth of $n_{y,k}$~~ we compute

$$\Delta t\,(\mathbf{q}\boldsymbol{q}_i(t_j,(\mathbf{y}\boldsymbol{y}_i)_{i=1}^{n_y},\mathbf{u}\boldsymbol{u},(\mathbf{b}\boldsymbol{b}_i)_{i=1}^{n_b},(\mathbf{d}\boldsymbol{d}_i)_{i=1}^{n_d}))_{i=1}^{n_y}\,. \tag{9}$$

Here, ~~$(\mathbf{y}_i)_{i=1}^{n}$~~ $(\boldsymbol{y}_i)_{i=1}^{n}$ is an input array of ~~n profiles, $\mathbf{u}$ a vector of m parameters, $(\mathbf{b}_i)_{i=1}^{n_b}$~~ n_y profiles, each with a length or depth of $n_{x,k}$, $\boldsymbol{u}$ a vector of n_u parameters, $(\boldsymbol{b}_i)_{i=1}^{n_b}$ a vector of n_b boundary data values and ~~$(\mathbf{d}_i)_{i=1}^{n_d}$~~ $(\boldsymbol{d}_i)_{i=1}^{n_d}$ an input array of n_d domain data profiles. Both inputs are regarded as already interpolated. The result is stored in the the output array ~~$(\mathbf{q}_i)_{i=1}^{n}$~~ $(\mathbf{q}_i)_{i=1}^{n_y}$ that consist of ~~n~~ n_y profiles as well. Formally, the tracer model is scaled with the (ocean) time step from the outside. However, we integrate Δt into the interface as a concession to the actual practice, where the time step is often refined within the tracer model implementation (cf. Kriest et al., 2010). Consequently, the responsibility to scale the result before returning it back to the transport driver software rests with the model implementer.

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

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

6 ~~Periodic solution~~

~~With those two building blocks, a model evaluation for a given parameter set $\mathbf{u}\in\mathbb{R}^m$ is a calculation of an annual periodic state that solves Equation with $y(t+1)=y(t)$ for every $t\in[0,1[$. This continuous solution translates after a spacial and temporal discretization to a sequence of states $(\mathbf{y}_j)_{j=1}^{n_t}$ with~~

$$\phi(\mathbf{y}_1,\mathbf{u}) = \mathbf{y}_1,$$

~~where $\phi=\varphi_{n_t}\circ\cdots\circ\varphi_1$ is the mapping that integrates a given tracer concentration over a model year (cf. Equation). Hence, the initial state of the discrete solution that we seek is a fixed point of ϕ.~~

~~Generally, we permit the integration to start at any $t_0\in[0,1[$. Independently of this choice, by definition the initial state is always depicted by $\mathbf{y}_1$. Howsoever, we omit the time index in the following for clarity.~~

5.1 ~~Spin-up~~

~~In this context, assuming that ϕ is a contraction, a spin-up is a fixed point iteration (Plato, 2003, pp. 109). It consist of the recurrent application of ϕ on the result of the previous iteration step, i.e.~~

$$\mathbf{y}_{l+1} = \phi(\mathbf{y}_l,\mathbf{u}),$$

~~where $l=1,\ldots,n_l$ is the model year index, n_l is the overall number of model years and $\mathbf{y}_l$ denotes the initial state of the lth model year. It can be understood as the propagation of the overall initial state over (typically) thousands of model years in order to reach an equilibrium (cf. Bernsen et al., 2008).~~

5.1 ~~Inexact Jacobian-free Newton-Krylov~~

~~On the other hand, Equation can be transformed into a zero finding problem on which Newton's method can be applied (cf. Kelley, 2003; Bernsen et al., 2008). For this purpose, we define $F(\mathbf{y},\mathbf{u})=\mathbf{y}-\phi(\mathbf{y},\mathbf{u})$ and solve $F(\mathbf{y},\mathbf{u})=0$ for a given parameter set $\mathbf{u}$. However, we omit the dependency of F on $\mathbf{u}$ in the following for clarity.~~

~~Using a Newton iteration in every step we solve~~

$$F'(\mathbf{y}_k)\,\mathbf{s}_k = -F(\mathbf{y}_k)\,,$$

~~where $k=1,\ldots$ is the Newton step index, F' denotes the Jacobian of F and $\mathbf{s}_k$ is the state update to find that is used to form the next iterate, i.e. $\mathbf{y}_{k+1}=\mathbf{y}_k+\mathbf{s}_k$. For this, the right-hand-side of Equation is computed first, which basically corresponds to *one* application of ϕ.~~

To solve the system of linear equations for a fixed k we use a Krylov subspace approach. It is a nested iteration to construct successive approximations that converge to the sought solution. These solvers require only the *result* of a matrix-vector product to proceed. Here, we choose the generalized minimal residual method (Saad and Schultz, 1986, GMRES), which is implemented as part of the linear solver suite in PETSc. In this context, the notion *Jacobian-free* refers to the fact that during the solving process the result of the Jacobian-vector product is approximated by a forward finite difference quotient, i.e.

$$F'(\mathbf{y}_k)\,\mathbf{s}_{k,l} \approx \frac{F(\mathbf{y}_k+\delta\,\mathbf{s}_{k,l})-F(\mathbf{y}_k)}{\delta},$$

where $l = 1,\ldots$ is the Krylov sub-index. The scaling parameter $\delta \in \mathbb{R}$ is chosen automatically as a function of $\mathbf{y}$ and $\mathbf{s}$ (cf. Balay et al., 2012a).

Within the inner loop the initial guess for the state update is always a vector of zeros, i.e. $\mathbf{s}_{k,1} = 0$ for every k. Thus, no computation is required for the first step and the initial Krylov residual is exactly the Newton residual, i.e. $F'(\mathbf{y}_k)\,\mathbf{s}_{k,1} + F(\mathbf{y}_k) = F(\mathbf{y}_k)$. Consequently, we overlay both points in a convergence plot. However, for the following iterations F must be evaluated at $\mathbf{y}^k + \delta\mathbf{s}_{k,l}$ to approximate $F'(\mathbf{y}_k)\,\mathbf{s}_{k,l}$. Here, again every evaluation is associated with *one* model year.

5.1 Convergence

Assuming there exist a unique solution of Equation , it can be found in a subspace of the Cartesian product of L^2 spaces over the time and space domain, i.e. $L^2(I\times\Omega)^n$ (cf. Evans, 1998, pp. 500). This space is equipped with the following (squared) norm

$$\|y\|^2_{L^2(I\times\Omega)^n} = \sum_{i=1}^{n} \int_I \int_\Omega |y_i(t,x)|^2\,dx\,dt\,.$$

We denote the discrete counterpart by

$$\|\boldsymbol{y}\|^2_{2,I\times\Omega} = \sum_{i=1}^{n}\sum_{j=1}^{n_t} \Delta t \sum_{k=1}^{n_y} w_k\,|y_{i,j,k}|^2,$$

where w_k is the *relative* volume of the partial grid box Ω_k, assuming the domain is scaled to a unit cube. Here, we use Δt instead of Δt_j due to the equidistant temporal resolution. In general, we omit the designation of the Cartesian product by the n in the norm notation for clarity.

However, the usage of the above norm involves the whole trajectory of all tracers and is thus expensive to compute. We mostly test for convergence by using an unweighted norm that only compares the initial states of consecutive model years. For a fixed time index j we then denote

$$\|\boldsymbol{y}\|_2^2 = \sum_{i=1}^{n}\sum_{k=1}^{n_y} |y_{i,j,k}|^2\,.$$

5.0.1 Spin-up

The difference between consecutive iterates is determined for a model year index $l = 2,\ldots,n_l$ as

$$\varepsilon_l = \|\mathbf{y}_l - \mathbf{y}_{l-1}\|_2\,.$$

The spin-up solver is easy to operate. The user can either set a tolerance ε that should be reached or a number of model years n_l that the initial state should be spun-up for. If both are set, the iteration stops at what is reached first.

5.0.1 Newton-Krylov

The Newton-Krylov solver is a more sophisticated approach than a spin-up. Various settings can be used to control the solving process. This is shown in more detail in Section 7.5, where results of numerical experiments are presented for a simple biogeochemical model.

In a convergence plot, every Newton step k is associated with the evaluation of *one* model year and the corresponding value is the norm of this so-called Newton residual, i.e. $\|F(\mathbf{y}_k)\|_2$. For the inner Krylov index l, every approximation of the Jacobian-vector product is again associated with *one* model year and the depicted value in a plot is the norm of the Krylov residual, i.e. $\|F'(\mathbf{y}_k)\,\mathbf{s}_{k,l} + F(\mathbf{y}_k)\|_2$.

The number of inner iterations per Newton step depends on the specified tolerance for the Krylov residual. For this, we use an already implemented convergence control based on a technique described by Eisenstat and Walker (1996). The inner tolerance is set in relation to the Newton residual and the solver proceeds until

$$\|F'(\mathbf{y}_k)\,\mathbf{s}_{k,l} + F(\mathbf{y}_k)\|_2 \leq \eta_k\,\|F(\mathbf{y}_k)\|_2$$

holds. This *inexact* approach avoids the so-called over-solving and decreases, especially in the beginning, the number of evaluations of F. The scaling factor η_k is determined from former Newton residuals as

$$\eta_k = \gamma\left(\frac{\|F(\mathbf{y}_k)\|_2}{\|F(\mathbf{y}_{k-1})\|_2}\right)^{\alpha}$$

with values set by default to $\eta_1 = 0.3$, $\gamma = 1$ and $\alpha = (1+\sqrt{5})/2$.

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 7.1.1). 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.

6.1 Layers

The implementation is structured in layers according to which the source files are named. A schematic is shown in Figure 11. 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 6.4) and to compute interpolation factors and indices (cf. Section 6.3) 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 6.2).

The next ~~both~~ 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 6.4). 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 6.5).

The next layer is the *time stepping* layer, where the main integration routine ϕ is located (cf. Algorithm 3). 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 ~~layer~~ calls to initialization respectively finalization routines are ~~combined as one call within~~ 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.

6.2 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~~ *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 1. Its theoretical and actual performance is discussed in Section 7.4 where we show results of speedup tests that we performed on two different hardware architectures.

6.3 Interpolation

The 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 number of files in each set is arbitrary.

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 6.5.

Concerning the boundary and domain forcing, we denote the data files by ~~$((\mathbf{b}_{i,j})_{j=1}^{n_{b,i}})_{i=1}^{n_b}$ and $((\mathbf{d}_{i,j})_{j=1}^{n_{d,i}})_{i=1}^{n_d}$~~ $((\boldsymbol{b}_{i,j})_{j=1}^{n_{b,i}})_{i=1}^{n_b}$ 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 per model year is generally much higher than the number of available data files. Thus, the matrices and vectors are ~~linearly~~ *linearly* interpolated to the current time step during the iteration. The 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 intervals. The appropriate weights and indices are computed on the fly using Algorithm 2. 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.

6.4 Biogeochemical model step

During a simulation the `BGCStep` routine in Algorithm 4 is responsible for the evaluation of the biogeochemical model. For this, the boundary and the domain data must be interpolated first. Here, for every index i and the corresponding boundary data set ~~$(\mathbf{b}_{i,j})_{j=1}^{n_{b,i}}$~~ $(\boldsymbol{b}_{i,j})_{j=1}^{n_{b,i}}$ we compute the appropriate weights α, β as well as indices j_α, j_β and form the

linear combination as

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

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

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

Technically, we use the PETSc routines `VecCopy`, `VecScale` and `VecAXPY` for this purpose, which is analogous to the interpolation of the transport matrices in Section 6.5.

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.

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

where $\underline{\mathbf{y}}_{i,k}$ $\boldsymbol{y}_{i,k}$ 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. Again, we group the domain data profiles by their profile index k, i.e.

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

where $\underline{\mathbf{d}}_{i,k}$ $\boldsymbol{d}_{i,k}$ 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.

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

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 (9). 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.

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

In contrast to the interpolation of vectors, and generally to all vector operations, each of the matrix operations has a significant impact on the computational time. In Section 7.3 we present results from profiling experiments that show detailed information about the time usage of each operation.

7 Results

In this section, we present results from numerical experiments to verify the software. At first, we We use the introduced interface to couple the transport matrix driver with a well investigated biogeochemical model implementation. We compare the simulation results with others and suite of biogeochemical models. We inspect the convergence behavior of both solvers included. Subsequently, we perform speed-up tests to analyze the implemented load distribution. A profiling of the main parts of the algorithm complements the initial verification.

Subsequent, we perform speed-up tests to analyze the implemented load distribution and compare it with the TMM. We continue by investigating the convergence control settings of the Newton-Krylov solver and examine the solver's behavior within parameter bounds. We finally present results from optimization runs against a reference solution.

7.1 Setup

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

7.1.1 ModelModels

In order to test our interface, we decide to couple an couple an N, N-DOP, NP-DOP, NPZ-DOP, NPZD-DOP model hierarchy and an *original* implementation of a biogeochemical model that is to the transport driver. The former is implemented from scratch for this purpose. The equations are shown in Appendix A. 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). ~~It has been widely investigated, which gives us the possibility to easily compare our results to those published by others. Moreover, we assume the model is correctly implemented. In particular, several experiments performed in (Kriest et al., 2010) and (Kriest et al., 2012) are based on its (slightly modified) source code.~~

~~The model comprises five biogeochemical variables, namely dissolved inorganic carbon (DIC), alkalinity (ALK), phosphate (PO4), dissolved organic phosphorous (DOP) and oxygen (O2). In fact, we will use just PO4 and DOP here since the concentrations of DIC, ALK and O2 are derived from those two. The model introduces seven parameters (cf. Table 19).~~ 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 ~~follow the introduced~~ use a convention for the directory structure to fit seamlessly into ~~the~~ an automatic compile scheme. Within the `model` directory of the `model` repository we create a folder ~~named~~ that is named after the biogeochemical model, i.e. `MITgcm-PO4-DOP` ~~.~~ for instance. ~~We implement a model wrapper for the original source code and store it in a~~ Within this directory we store the source code file named `model.F` ~~within that folder~~. 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

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

for instance and such create an executable named

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

that we use for *all* the following experiments. Specific settings will be provided via option files.

7.1.2 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 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.

The ~~used MITgcm-PO4-DOP model determines the number of tracers to $n = 2$ and the parameter count to $m = 7$ (cf. Table 19). The components of the combined tracer vector are $\mathbf{y}_{PO4}$ and accordingly $\mathbf{y}_{DOP}$, i.e. $\mathbf{y} = (\mathbf{y}_{PO4}, \mathbf{y}_{DOP})$. 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. ~~For more information, an algorithm of a very similar model can be found in Siewertsen et al. (2013). Finally,~~ 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. ~~We initialize the variables with global mean concentrations of $\mathbf{y}_{0,PO4} = 2.17\ \mathrm{mmol\,P/m^3}$ and $\mathbf{y}_{0,DOP} = 0.0001\ \mathrm{mmol\,P/m^3}$, respectively.~~

7.2 Solver

We begin our verification by computing a ~~reference solution for the parameter set $\mathbf{u}_d$ that is depicted in Table 19.~~ steady annual cycle for every model with both solvers. ~~Both solvers are started with the same initial configuration.~~

Regarding the spin-up, we set no tolerance and let the solver iterate for 10,000 model years~~, despite the fact that usually 3,000 are regarded as sufficient (cf. Bernsen et al., 2008).~~ . 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 overall absolute tolerance is at 10^{-8} and the maximum number of inner iterations is 10,000.

~~Figure ?? shows the~~ The parameter values we use for the MITgcm-PO4-DOP model are depicted in Table 18 and named $\boldsymbol{u}_d$ therein. Table 19 depicts 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

$2.17\,\mathrm{m\,mol\,P\,m^{-3}}$ for N or PO4 and $0.0001\,\mathrm{m\,mol\,P\,m^{-3}}$ for the other tracers.

For the MITgcm-PO4-DOP model a comparison of the convergence towards a ~~periodic steady state~~steady annual cycle for both solvers is shown in Figure 13. ~~Both solver obviously converge towards the same solution~~We observe that the solutions converge to the same difference in between consecutive iterations. ~~The difference is generally measured using the unweighted norm of initial states consecutive model years~~Moreover, Table 16 shows the difference between both solutions in Euclidean norm. Additionally, Figure 19 depicts the difference between both solutions for the surface layer. Except for the numerical error, both solvers obviously compute the same solution.

Figures 14 and 15 show the convergence behavior of both solvers for the N respectively N-DOP model. There is no essential difference in comparison to the MITgcm-PO4-DOP model. An inspection of the surface Figures 110 and 111 confirms this impression. There is no peculiarity shown in Table 16 either.

However, for the NP-DOP model Figure 16 shows a different behavior of the Newton-Krylov solver at the end of the solution process. ~~Additionally, every 100 years we computed the weighted norm between whole trajectories for comparison.~~ A closer inspection reveals a peak every 30 model years, which obviously results from the settings of inner solver, where GMRES is set to perform a restart every 30 years by default. Surface Figure 112 and Table 16, however, do not indicate any effect on the solution.

The NPZ-DOP and NPZD-DOP models show a different behavior regarding the Newton solver. For both models, the solver does not converge with default settings as shown in Figure 17 (top) and Figure 18 (top). It can be seen that the reduction of the residual per step is quite low, which results in a huge number of iterations. Here, the solver was stopped after 50 iterations (the default), which already is a high number for Newton's method. The reason is that convergence of the method – even in its so-called globalized or damped version used here – still may depend on the initial guess y^0. We used a different one, which was successful for the NPZD-DOP model, see Figure 18 (middle). For the NPZ-DOP model, it still was not, see Figure 17 (middle).

However, a second and much easier way to achieve convergence can be deduced already from Figure 17 (top) and Figure 18 (top). The stopping criterion of the inner iterations of the Newton solver is less restrictive if the last Newton iteration was not very successful, which is obviously the case here. The number of inner iterations and thus the accuracy of the Newton direction is improved when the inner criterion (10) is sharpened, thus somehow contradicting the idea formulated in Eisenstat and Walker (1996). This can be easily achieved by decreasing γ, here to $\gamma = 0.3$. This tuning now led to convergence, see Figure 17 (bottom) and Figure 18 (bottom). With this settings, the respective solutions are the same as the ones obtained by the spin-up, when numerical errors are neglected (see Figures 113 and 114). This is also confirmed by evaluating the differences in the norm, see Table 16.

Overall, we observe that the Newton-Krylov solver does not reach the default tolerance and iterates unnecessarily for 10,000 model years within the last Newton step. Thus, we limit the inner Krylov iterations to 200 in the following experiments. Moreover, for further investigations with the MITgcm-PO4-DOP model we change the convergence settings to get rid of the over-solving that we observe at the beginning. Referring to this, more detailed experiments are presented in Section 7.5.

~~Nevertheless, the results resemble the observational data taken from the World Ocean Database (Boyer et al., 2013), which were mapped onto a $2.8125°$ grid and interpolated in space and time for comparison. Figure ?? shows the concentration of phosphate within the first layer. Here, the data is shifted to show Greenwich ($0°$) at the center. Moreover, Figure ?? depicts slices through the Pacific, Atlantic and Indian. Consequently, we assume the coupling of the biogeochemical model to the transport driver was successful.~~

7.3 Profiling

~~Confident that the compiled executable produces correct results,~~ In following two sections we investigate some technical aspects of the implementation more closely. First of all, we are interested in the distribution of the computational time among the main operations of a model year.

For this, we perform a *profiled* sequential run for each model at which we iterate for 10 model years. The analysis of the profiling results is shown in ~~Figure ??~~Figures 117 - 115. ~~We~~ Regarding the MITgcm-PO4-DOP model for instance, we observe that the biogeochemical model takes up 40% of the computational time. The interpolation of matrices (`MatCopy`, `MatScale` and `MatAXPY`) amounts to approximately a third. The matrix vector multiplication (`MatMult`) takes up a quarter of the computations and all other operations amount to ~~1.5%~~0.5%.

Moreover, we recognize that the more tracers are involved the more the matrix vector multiplication becomes dominant. For the N model it takes up $19,8\%$ of the computational time, whereas for the NPZD-DOP model the `MatMult` operation amounts to $56,7\%$. The possible implications are discussed in Section 8.

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.

7.4 Speed-up

~~Regarding the solver experiment, we have chosen the number of processes as such that the computations become feasible.~~ In this section, we investigate the performance of the load balancing algorithm in detail ~~.~~ and compare the results with the parallel performance of the TMM. We compile both drivers with the same biogeochemical model. For this purpose we choose MITgcm-PO4-DOP since it is part of the TMM as well and, consequently, we have the same setup.

We run ~~tests on two different hardware platforms. The first hardware is an (older) AMD® Barcelona architecture that consists of Opteron® 2352 CPUs with 4 cores running at 2.1 GHz~~the tests on a hardware that located at the computing center of Kiel University. ~~The second~~ It is an Intel® Sandy Bridge EP architecture with Intel Xeon® E5-2670 CPUs that consist of ~~8~~ 16 cores running at 2.6 GHz. ~~Both are integrated into a computer cluster located at the computing center of the university of Kiel.~~

~~On each hardware,~~ Regarding our implementation we perform 10 tests ~~with respect to a specific number of processes. Regarding the AMD Barcelona hardware we use 1 to 184 cores, on the Intel Sandy Bridge EP hardware each simulation run is performed~~ using 1 to 256 cores. Each test consists of ~~running simulations~~ a simulation run of three model years, at which each year is timed separately. For the TMM we use 1 to 192 cores and run 5 tests on each core. 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 ~~smallest measured time of these 30 tests, i.e. the best performance per number of processes.~~

~~All~~ minimum timings for a specific number of cores. Moreover, all timings are related to the timing of a sequential run. ~~The absolute sequential minimum timings are $t_1 = 646.592\,\mathrm{s}$ (AMD) and $t_1 = 153.038\,\mathrm{s}$ (Intel), respectively.~~ For a set of measured computational times $(t_i)_{i=1}^{N}$ with ~~$N = 184$~~ $N = 192$ or $N = 256$ we calculate the speedup as $s_i = t_1/t_i$ and the efficiency as $e_i = 100 * s_i/i$.

Additionally, referring to the implemented load distribution (cf. Section 6.2), we compute the best possible ratio between a sequential and a parallel run. For all number of processes, i.e. $i = 1, \ldots, 260$, we compute the load distribution using Algorithm 1 and retrieve the maximum (local) length $n_{i,max}$. For the speed-up we divide the vector length by this value, i.e. $s_i = n_y/n_{i,max}$, and for the efficiency we again calculate $e_i = 100 * s_i/i$.

Figure 118 depicts the ideal, ~~best possible~~ theoretical and actual speedup respectively efficiency. Regarding the implemented load distribution a good (theoretical) performance over the whole range of processes can be observed. ~~However~~Moreover, we recognize that ~~on the AMD hardware~~ a parallel run ~~never reaches the theoretically possible speed-up. The best performance is achieved between 90 and 100 processes, at which the speed-up is at 70 and the efficiency slightly over 70%. Thereafter the speed-up remains the same but the efficiency decreases.~~

~~In contrast, a parallel run~~ of Metos3D on the Intel hardware reaches between 100 and 140 ~~processes~~ cores almost best performance. In this range the efficiency is about 95% and the speed-up nearly corresponds to the number of processes. ~~After that, the efficiency drops constantly as observed for the AMD architecture.~~ Indeed, the speed-up still rises to slightly over 160 but requires at least 200 processes to reach this factor.

In contrast, the performance of the TMM is poor. The efficiency drops from the beginning and a speedup higher than 40 is never reached. From 120 cores up Metos3D is at least 4 times faster. Interestingly, there is a significant drop in performance at the beginning ~~on both architectures~~for both drivers. ~~In particular, each hardware shows a different pattern.~~ The possible implications are shortly discussed in Section 8. However, since the results give us a good orientation anyway this effect is not investigated further. ~~Overall, as already indicated by the sequential runs, the Intel hardware is the obvious choice for subsequent experiments.~~

7.5 Convergence control

After a basic verification and a review of technical aspects of our implementation, we investigate the settings to control the convergence of the Newton-Krylov solver. Again, we use the MITgcm-PO4-DOP model only. Our intention is to eliminate the over-solving that we observe during the first 200 iterations in Figure ~~??~~13. 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 two is controlled by the γ and the α parameter depicted in Equation (10).

Hence, we compute the reference solution from Section 7.2 with different values of γ and α to investigate their influence on the convergence behavior. We set the overall tolerance to the measured difference of consecutive states after 3,000 model years of spin-up, i.e. approximately 9.0×10^{-4}. We let the value of γ vary from 0.5 to 1.0 in steps of 0.1 and α is chosen from 1.1 to 1.6 in steps of 0.1 as well. This is a total of 36 model evaluations.

Figure 119 depicts the required model years and Newton steps as a function of γ and α. We observe that the overall number of years decreases, as both parameters tend to 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 residual is negligible in comparison to the simulation of a 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 after approximately 450 model years, which is significantly less than 600

if using the default settings. Thus, we use these values for the next ~~experiments~~experiment.

7.6 Parameter samples

~~As mentioned in the introduction, one of the strategies to accelerate the computation of periodic steady-states was to utilize a Newton approach. After an initial verification, we are confident that the Newton-Krylov solver is working correctly and, with optimal settings, at least 6 times faster than the spin-up (fixed point iteration).~~

~~However, until~~ Until now we solved the given model equations for ~~the reference~~parameter set $\mathbf{u}_d$ ~~one (reference)~~ parameter set only. During an optimization a solution must be computed for various parameter sets. Thus, we perform the next experiments in order to study the solver's behavior with regard to other model parameters. Again, we use the MITgcm-PO4-DOP model only. For this purpose, using the MATLAB® routine `lhsdesign`, we create 100 Latin Hypercube (cf. McKay et al., 1979) samples within the bounds that are depicted in Table ~~19.~~18. We set the overall tolerance again to a value that is comparable with 3,000 spin-up iterations and let the Newton solver compute a solution for each parameter sample

Figure 120 shows histograms of the total number of model years respectively Newton steps required to solve the model equations. We observe that most computations converge in between 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. Moreover, we recognize some outliers in both graphs. Nevertheless, all started model evaluation converged towards a solution within the desired tolerance. ~~Thus prepared, we carry out the last experiment.~~

7.7 ~~Twin Experiment~~

8 Conclusions

~~Finally, after a verification of the spin-up and the Newton approach, we perform a twin experiment with each solver. We separately compute a reference solution with specific settings and start an optimization run (using the same settings) against it~~We designed and implemented a simulation framework for the computation of steady annual cycles for a general class of marine ecosystem models in 3-D, driven by pre-computed transport matrices in an off-line mode. ~~Regarding the~~ The framework allows computation of the steady cycle(s) by a spin-up ~~we let the solver iterate for 3,000 model years and set no tolerance once again~~or a globalized Newton method. The ~~Newton solver is set up as described in Section 7.5~~software is completely realized as (or using available) open source code.

We ~~consider the following optimization problem:~~

$$\min_{\mathbf{u}\in U} J(\mathbf{u}),$$

introduced a software interface for water column-based biogeochemical models. ~~where~~

$$J(\mathbf{u}) = \tfrac{1}{2}||\mathbf{y}(\mathbf{u}) - \mathbf{y}_d||^2_{2,I\times\Omega}$$

~~and the admissible set is defined as~~

$$U = \{\mathbf{u}\in\mathbb{R}^m : \mathbf{b}_l \le \mathbf{u} \le \mathbf{b}_u\}.$$

~~Here, $\mathbf{y}_d = \mathbf{y}(\mathbf{u}_d)$ is the reference solution computed before and $\mathbf{b}_l$ respectively $\mathbf{b}_u$ are the lower and upper bounds we impose during the optimization~~On one hand, we showed the applicability and flexibility of this interface by coupling the biogeochemical component used in the MITgcm general circulation model to the simulation framework. ~~The norm is computed using the whole trajectory and both optimization runs are started with $\mathbf{u}_0$ (cf. Table 19)~~ On the other hand, we coupled own implementations of five other biogeochemical models (also used in Kriest et al. (2010)) with different complexity to show the interface's generality. Their source code is also available within the software, and may serve as templates for implementation or adaption of other models.

~~To solve the problem we use MATLAB's `fmincon` routine for constraint nonlinear optimization, where we set the algorithm to active-set (cf. Nocedal and Wright, 2000, pp. 308)~~We implemented a transient solver based on the transport matrix approach, where all matrix operations and the evaluation of the biogeochemical models are performed with spatial parallelization via MPI using the PETSc library. ~~It is~~ The needed transport matrices are directly available and require no pre-processing.

We realized both a ~~quasi-Newton approach, at which the inverse of the Hessian is approximated using Broyden's method (cf. Dennis and Schnabel, 1996, pp. 169)~~spin-up (or fixed-point iteration) and a globalized Newton solver for the computation of steady cycles. ~~In both twin experiments we approximate the gradients with forward finite differences and a step size that equals the square root of the machine precision.~~ We compared these solvers and made the following observations: ~~Regarding the Newton solver, one additional experiment is carried out with a relative step size of 10^{-4}~~Both deliver the same results (up to a reasonable precision) on convergence.

~~Figure ?? shows the result of the optimization run(s) using the~~ The spin-up ~~solver~~converges with standard sets of parameters, taken from Kriest et al. (2010), for equally distributed values for each tracer. ~~Due to the time limitation of the used batch system, we had to restart the first run, which has been stopped after 200 hours~~The Newton solver showed the same behavior for the four models of lower complexity.

~~Thereby, the approximation information about the Hessian was lost, which explains the different path that is taken by the second run.~~ For the other two, it did not converge with the standard setting of its parameters and the mentioned initial distribution of tracers. ~~However, we observe a convergence of the parameters towards their reference values. At the last optimization step they are $\mathbf{u} = (0.499$~~For both of these two more complex models, convergence was achieved by increasing the number of inner iterations in the Newton solver, ~~$2.016, 0.670, 0.502, 30.461, 0.019, 0.858).$~~ which is realized by decreasing the parameter γ in (10). ~~Here,~~ For one of these models, the ~~values are round off to three decimal places.~~ same could be achieved by choosing a different initial guess.

~~Figure ?? depicts the attempt to minimize the cost function using the Newton solverfor model evaluation. Both optimization runs finish because the predicted change in the objective function is less than 10^{-6}, which is~~ Concerning performance, the ~~default value of the function tolerance. They need about 430 respectively 470 model evaluations, which corresponds to slightly more than 210,000 overall model years each. However, both attempts obviously fail to identify the reference parameter set~~Newton solver was about 6 times faster for all models. ~~Here, based on the two experiments, a detailed analysis is hardly possible.~~ It can be concluded that the Newton method requires more thorough solver parameter setting for complex models, but then is superior in any case, at least for the considered parameter sets. ~~They provide only first clues (cf. Section 8).~~

~~In order to fundamentally tackle the problem of parameter identification for marine ecosystem models in 3-D, we introduced a general biogeochemical programming interface that fits into the optimization context. Moreover, we implemented a comprehensive parallel solver software for periodic steady-states that uses the interface to couple marine ecosystem modelsto a transport matrix driver.~~

~~We validated the new implementation using a simple biogeochemical model knowing full well that the model is too simple for the intended purpose. Referring to this, preliminary experiments with more complex descriptions of the marine ecosystem, as the O2-NPZD-DOP model used by Kriest et al. (2010) for instance, did not provide new insights regarding a basic verification. On the contrary, they further complicated the investigation and were thus not described here.~~

We ~~primary focused on the technical aspects of the software, the employed solvers and, finally, the usage of each solver for parameter identification~~studied the dependency of the Newton performance with respect to the two solver parameters α, γ in (10) for one exemplary model. ~~Here, we have seen how useful the inherited profiling capability can be to access the computational complexity of a new model implementation~~With an optimal choice derived from these experiments (for one model parameter set), we then investigated the dependency of the needed Newton iterations and overall model years for 100 latin hypercube model parameter samples. ~~Moreover, the performed speed-up tests revealed that a parallel hardware needs to be carefully inspected before it is used for numerical experiments. For instance, using the Intel architecture, it would unfavorable to split 128 available processes into 8 separate experiments. Despite a perfectly working load balancing this would result in only 50% of the possible performance.~~

~~Furthermore, regarding the Newton solver, model evaluations with different parameter samples and control settings confirmed what has already been stated by Kelley (2003) for instance~~This test is important for the usability of the Newton method for example in a optimization run where model parameters are varied by the optimizer. ~~The PETSc library provides a flexible and robust solver implementation that, in our case, solves the given nonlinear equations at least 6 times~~ It turned out that there is a variance in the needed steps and thus the overall effort, but that there are no extreme outliers. We conclude that the Newton method – at least for this model – is appropriate for optimization, and faster than the ~~fixed point iteration.~~ usually robust spin-up.

~~However, concerning the twin experiments, we must recognize that both solving approaches have their own specific difficulties with regard to a derivative based black-box optimization~~We further analyzed the proportions in time that the different pieces of the simulation in one model year need. ~~Note thatthe chosen optimization approach was somehow "natural"~~It turned out that, with increasing number of tracers, the matrix-vector operations dominate and thus have the most potential for further performance tuning. This ~~work focused on the computation of periodic steady-states, i.e. mere model evaluations, and we used a model that is smooth enough, i.e. for which derivative information is available~~is despite the fact that the transport operator for every tracer is the same. ~~The intention was to avoid a whole survey of optimization methods including a variety of derivative-free approaches (cf. Rios and Sahinidis, 2013).~~

~~Howsoever, although a finite differences approximation of gradients works fine with the fixed point iteration, it is computationally still too complex. Overall, more than 951,000 model years were simulated during the spin-up twin experiment. The approach may be easy to realize, but it clearly consumes to many computational resources. At least, it shows that the model parameters can be recognized.~~

~~Here, the obvious idea would be to take coarser time steps as implied by (Khatiwala, 2007).~~ However, ~~new transport matrices need to constructed for this purpose.~~ it still has to be evaluated, whose effort is proportional to the number of tracers in the model. ~~Indeed, the appropriate scripts are provided in the `data` repository of Metos3D, but once again, not to further complicate a basic verification the approach was not discussed here~~In contrary, the biogeochemical

interactions in the nonlinear coupling terms q_j, which are mostly spatially local, become less performance-relevant.

~~Moreover, due to the fact that a coarser time step may lead to inaccurate results, a Newton solver was integrated into the software~~Finally, we implemented a load balancing that exploits the different depths of the water columns in the ocean that result in different lengths of the corresponding data vectors. ~~And, as it turned out, a model evaluation using a Newton approach is much faster~~With this balancing, a nearly optimal speed-up by spatial parallelization up to about a comparably high number of 128 processes was possible. ~~However, the employed optimizer apparently struggles with the approximation of gradients by finite differences using this solving approach. A closer inspection of the results reveals that the computed gradients differ from those using a fixed point iteration. Here, a separate investigation is necessary.~~ This is a huge difference to the performance with standard load balancing.

~~Furthermore, usually the employment of pre-conditioners must be taken into account, as has already been discussed by Khatiwala (2008)~~Summarizing, the presented software framework is an appropriate tool to be used in parameter optimization and model assessment runs. ~~Indeed, PETSc offers several own pre-conditioner implementations or at least the possibility to interact with the inner solver at the appropriate location~~It has high flexibility w.r.t. models and steady cycle solvers, offers improved parallel performance and can be easily combined with any optimization method. ~~However, none of the included PETSc pre-conditioner nor the presented approach by Khatiwala (2008) is matrix-free.~~ 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. ~~Thus, once again, in order to not further complicate the basic verification this has not been considered here.~~ Our results show that the parallelization effort is well-invested in the simulation itself.

~~Finally, we would like to note that introduced programming interface showed the expected flexibility with regard to a model coupling on source code level. Though, we realized a 1-D (water column) interface only, this is no restriction for future development. Moreover, preliminary experiments showed that, regarding Algorithmic Differentation, and an interface for a forward and/or reverse mode, can easily be derived.~~

9 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: 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 ~~version has been tagged as~~ versions of `simpack` and `model` are tagged as `v0.3.2`. The data is repository is at version `v0.2`. All experiments presented in this work were carried out using this ~~version~~versions. The associated material is stored in the ~~`verification`~~`2016-GMD-Metos3D` repository.

To install the software, the user should visit the homepage and follow the instructions. Whereas in the future an installation will always reflect the current state of development, the user can always invoke `git checkout ~~tags/~~v0.3.2` in the `simpack` ~~,~~ and `model` ~~,~~ repository as well as `git checkout v0.2` in the `data` ~~and verification repository , respectively,~~ repository to retrieve the ~~version~~ versions used in this work.

Appendix A: Model equations

The here presented N, N-DOP, NP-DOP, NPZ-DOP and NPZD-DOP model hierarchy is based on the descriptions used by Kriest et al. (2010). The introduced parameters are shown in Table 19.

A1 Short wave radiation

As mentioned Section 7.1.2, the short wave radiation for the *topmost* layer is deduced from the insolation that is computed on the fly using the formula of Paltridge and Platt (1976). Here, latitude ϕ and ice cover σ_{ice} data is required. We denote the computed value by $I_{SWR} = I_{SWR}(\phi, \sigma_{ice})$. For the lower layers their depths $(z_j)_{j=1}^{n_x}$ and heights $(dz_j)_{j=1}^{n_x}$ are required. Additionally, the attenuation of water is described by the coefficient k_w respectively the attenuation of phytoplankton (chlorophyll) by k_c.

A1.1 Implicit phytoplankton

For the N and the 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 $I'_j = \exp(-k_w\, dz_j/2)$, $I_k = \exp(-k_w\, dz_k)$ and j is the actual layer index.

A1.2 Explicit phytoplankton

For the NP-DOP, NPZ-DOP and NPZD-DOP model the short wave radiation is computed *with* phytoplankton, 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 $I'_{P,k} = \exp(-(k_w + k_c y_{P,k}) dz_k)$.

A2 N model

The simplest model consists of nutrients (N) only, i.e. $\boldsymbol{y} = (\boldsymbol{y}_N)$. Table A1 depicts the equation. The 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 $y^*_P = 0.0028\,\mathrm{mmol\,P/m^3}$. The N model introduces $n_u = 5$ parameters, where $\boldsymbol{u} = (k_w, \mu_P, K_N, K_I, b)$.

A3 N-DOP model

The N-DOP model consists of nutrients (N) and dissolved organic phosphorous (DOP), i.e. $\boldsymbol{y} = (\boldsymbol{y}_N, \boldsymbol{y}_{DOP})$. The computation of the biological uptake remains the same. Table A2 depicts the equations. The N-DOP model introduces $n_u = 7$ parameters, where $\boldsymbol{u} = (k_w, \mu_P, K_N, K_I, \sigma_{DOP}, \lambda_{DOP}, b)$.

A4 NP-DOP model

The NP-DOP consists of nutrients (N), phytoplankton (P) and dissolved organic phosphorous (DOP), i.e. $\boldsymbol{y} = (\boldsymbol{y}_N, \boldsymbol{y}_P, \boldsymbol{y}_{DOP})$. Here, the 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 of short wave radiation changes as well (see Section A1.2). Additionally, a quadratic loss term for phytoplankton is introduced and 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 $y^*_Z = 0.01\,\mathrm{mmol\,P/m^3}$. Table A3 depicts the equations. The NP-DOP model introduces $n_u = 13$ parameters, where $\boldsymbol{u} = (k_w, k_c, \mu_P, \mu_Z, K_N, K_P, K_I, \sigma_{DOP}, \lambda_P, \kappa_P, \lambda'_P, \lambda_{DOP}, b)$.

A5 NPZ-DOP model

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

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

Table A4 depicts the equations. The NPZ-DOP model introduces $n_u = 16$ parameters, where $\boldsymbol{u} = (k_w, k_c, \mu_P, \mu_Z, K_N, K_P, K_I, \sigma_Z, \sigma_{DOP}, \lambda_P, \lambda_Z, \kappa_Z, \lambda'_P, \lambda_Z, \lambda DOP, b)$.

A6 NPZD-DOP model

The NPZ-DOP consists of nutrients (N), phytoplankton (P) zooplankton (Z), detritus (D) and dissolved organic phosphorous (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 linear sinking speed is introduced for detritus. Table A5 depicts the equations. The NPZD-DOP model introduces $n_u = 16$ parameters, where $u = (k_w, k_c, \mu_P, \mu_Z, K_N, K_P, K_I, \sigma_Z, \sigma_{DOP}, \lambda_P, \lambda_Z, \kappa_Z, \lambda_P, \lambda_Z, \lambda_D, \lambda_{DOP}, a_D, b_D)$.

Acknowledgements. The authors would like to thank S. Khatiwala for providing support on the transport matrices and for providing 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. ~~At last, we thank our colleague Joscha Reimer for preparing the World Ocean Database data.~~ This work was partly funded by *The Future Ocean* cluster.

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.

Table A1. Equations for the N model with $E = f_P(y_N, I)$.

	Euphotic zone	Sinking
$q_N(y) =$	$-f_P(y_N, I)$	$+E\,dz_j\,\partial_z(z_k/z_j)^{-b}$

Table A2. Equations for the N-DOP model with $E = \bar{\sigma}_{DOP}\,f_P(y_N, I)$.

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

Table A3. Equations for the NP-DOP model with $E = \bar{\sigma}_{DOP}\,f_Z(y_P)$.

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

Table A4. Equations for the NPZ-DOP model with $E = \bar{\sigma}_{DOP}(\bar{\sigma}_Z\,f_Z(y_P, y_Z) + \lambda_P\,y_P + \kappa_Z\,y_Z^2)$.

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

Table A5. Equations for the NPZD-DOP model.

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

Boyer, T., Antonov, J., Baranova, O., Coleman, C., Garcia, H., Grodsky, A., Johnson, D., Locarnini, R., Mishonov, A., O'Brien, T., Paver, C., Reagan, J., Seidov, D., Smolyar, I., and Zweng, M.: World Ocean Database 2013, Tech. rep., NOAA Atlas NESDIS 72, Silver Spring, s. Levitus, Ed.; A. Mishonov, Technical Ed., 2013.

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.

Evans, L.: Partial Differential Equations, American Math. Society, Providence, Rhode Island, 1998.

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.: Transport Matrix Method Web page, http://www.ldeo.columbia.edu/%7Espk/Research/TMM/ (last access: 12 July 2013), 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.

Kriest, I., Oschlies, A., and Khatiwala, S.: Sensitivity analysis of simple global marine biogeochemical models, Global Biogeochemical Cycles, 26, doi:10.1029/2011GB004072, http://oceanrep.geomar.de/14320/, 2012.

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.

Nocedal, J. and Wright, S. J.: Numerical Optimization, Springer, New York, 2000.

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

Plato, R.: Concise numerical mathematics, 57, American Mathematical Soc., 2003.

Rios, L. M. and Sahinidis, N. V.: Derivative-free optimization: A review of algorithms and comparison of software implementations, Journal of Global Optimization, 56, 1247–1293, 2013.

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.

Temam, R.: Navier-Stokes Equations, North-Holland, Amsterdam, 1979.

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.

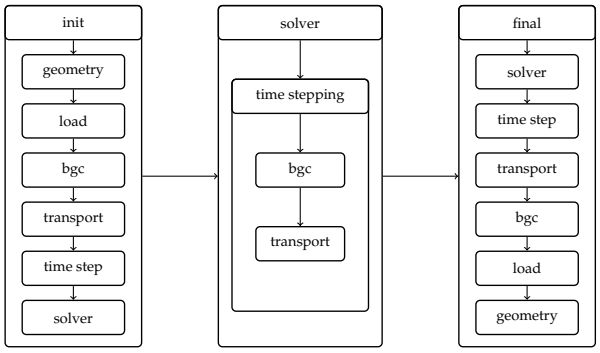

Figure 11. Schematic of the implementation structure of Metos3D.

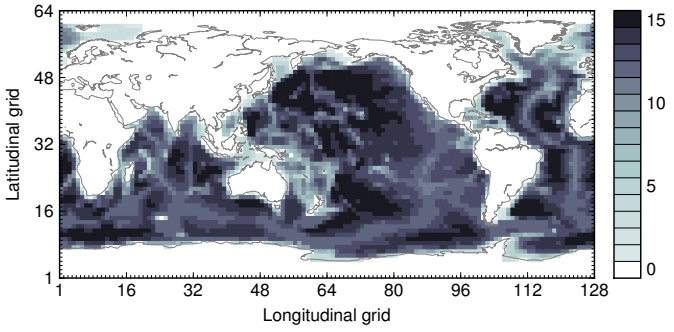

Figure 12. Land-sea mask (geometric data) of the used numerical model. Shown are the number of layers per grip point. Note that the Arctic has been filled in.

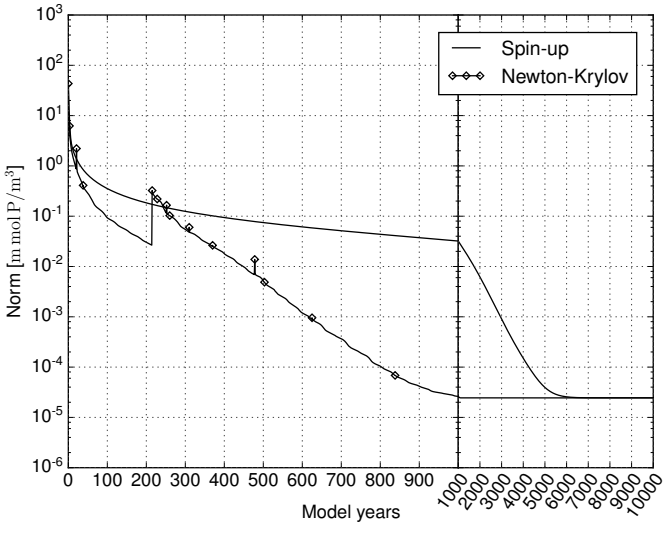

Figure 13. *MITgcm-PO4-DOP model:* Convergence towards an annual cycle. *Spin-up:* norm of difference between initial states of consecutive model years (solid line)~~and trajectories every hundred model years (dots with dashed line)~~. *Newton-Krylov:* residual norm at a Newton step (diamond) and norm of the GMRES residual during solving (solid line in-between).

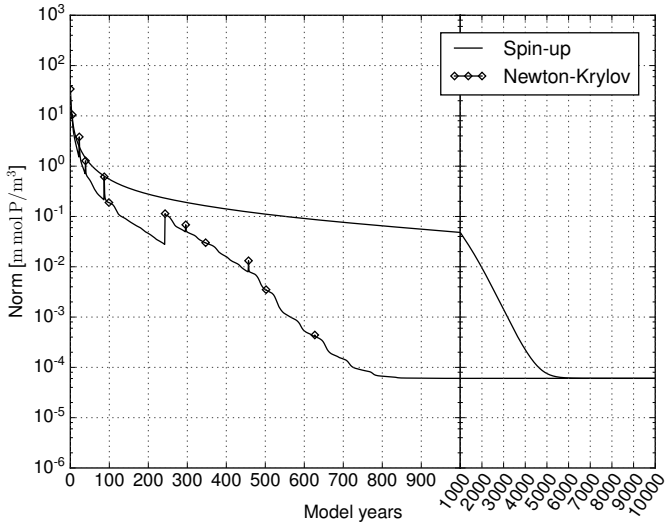

Figure 14. ~~Concentration of phosphate (y_{PO4}) at the first layer (0 – 50 m)~~ *N model:* Convergence towards an annual cycle using a spin-up and a Newton-Krylov solver. ~~*Left:* Shown is the initial state (1st of January, 00:00 am) of the converged~~

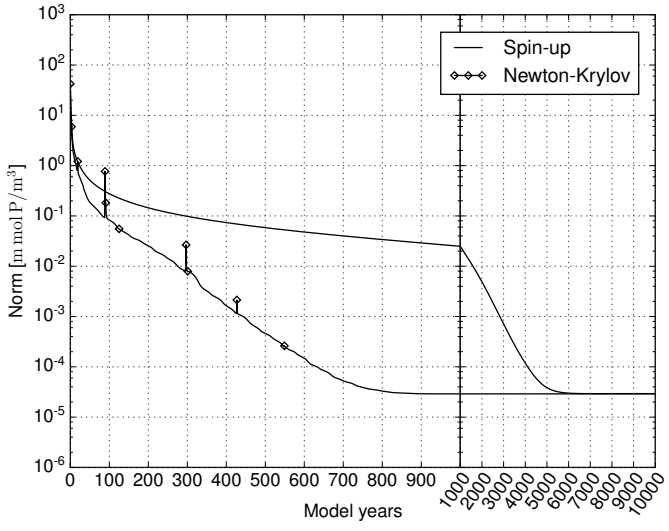

Figure 15. *N-DOP model:* Convergence towards an annual cycle ~~presented in Figure ??~~using a spin-up and a Newton-Krylov solver. ~~*Right:* Interpolated World Ocean Database observational data for the same point in time~~

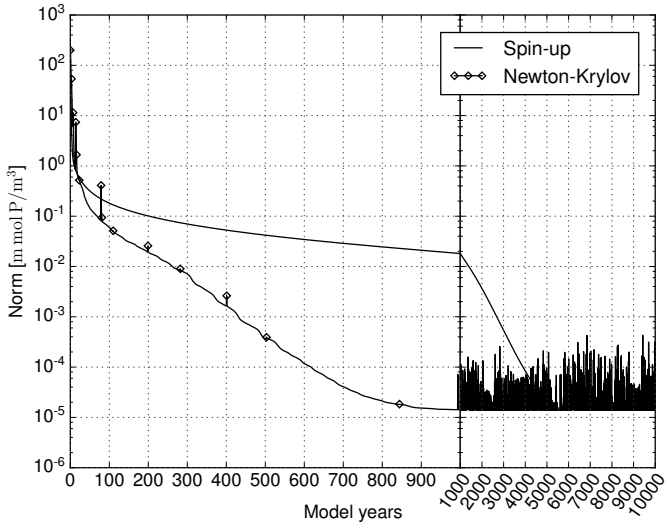

Figure 16. *NP-DOP model:* Convergence towards an annual cycle using a spin-up and a Newton-Krylov solver. ~~The dashed lines depict locations of slices shown in Figure ??.~~

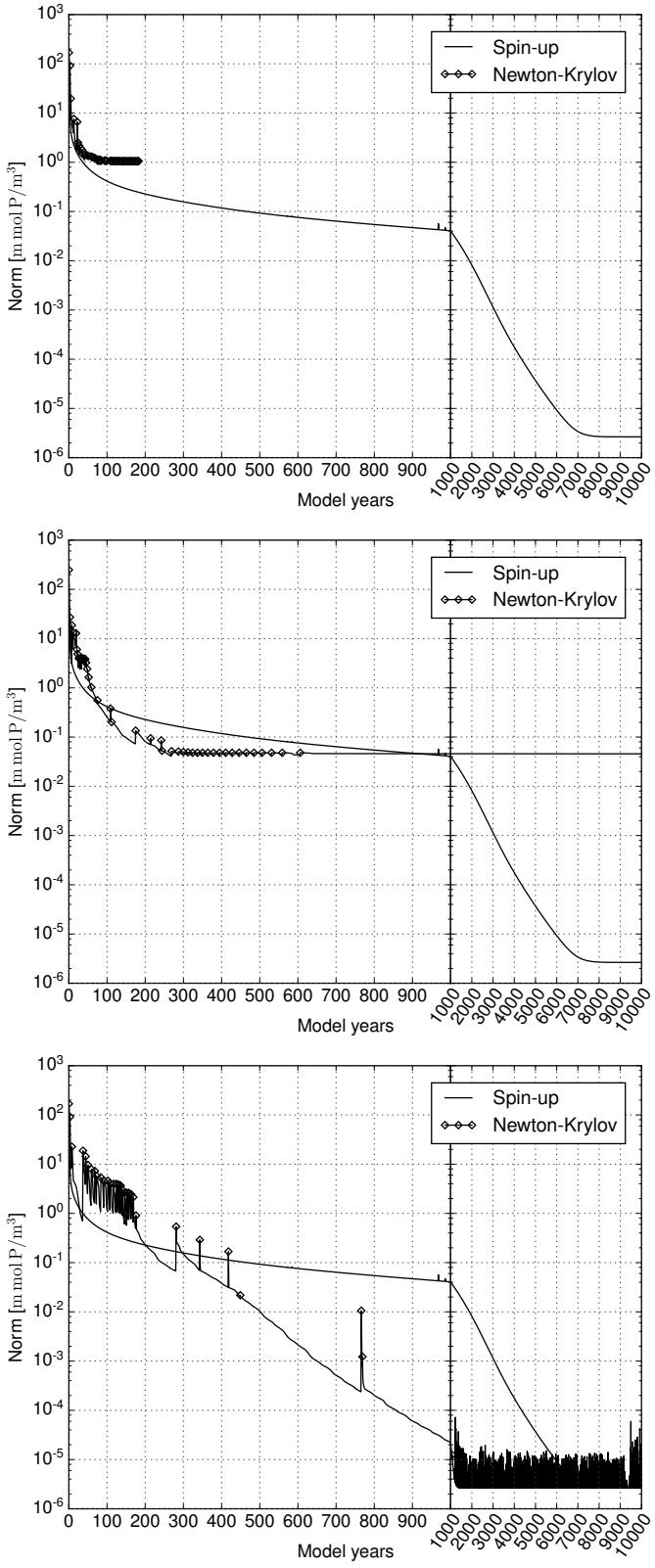

Figure 17. ~~Slices corresponding to Figure ??: the Pacific (153.2815° W), the Atlantic (29.53125° W)~~ *NPZ-DOP model:* Convergence towards an annual cycle using a spin-up and ~~the Indian (91.40625° E)~~a Newton-Krylov solver. ~~*Left:* Simulated tracer concentration~~*Top:* Default Newton-Krylov setting. ~~*Right:* Observational data from the World Ocean Database 2013.~~*Middle:* Changed initial value to $0.5425\ \mathrm{m\,mol\,P\,m^{-3}}$ for all tracers. *Bottom:* Changed inner accuracy to $\gamma = 0.3$.

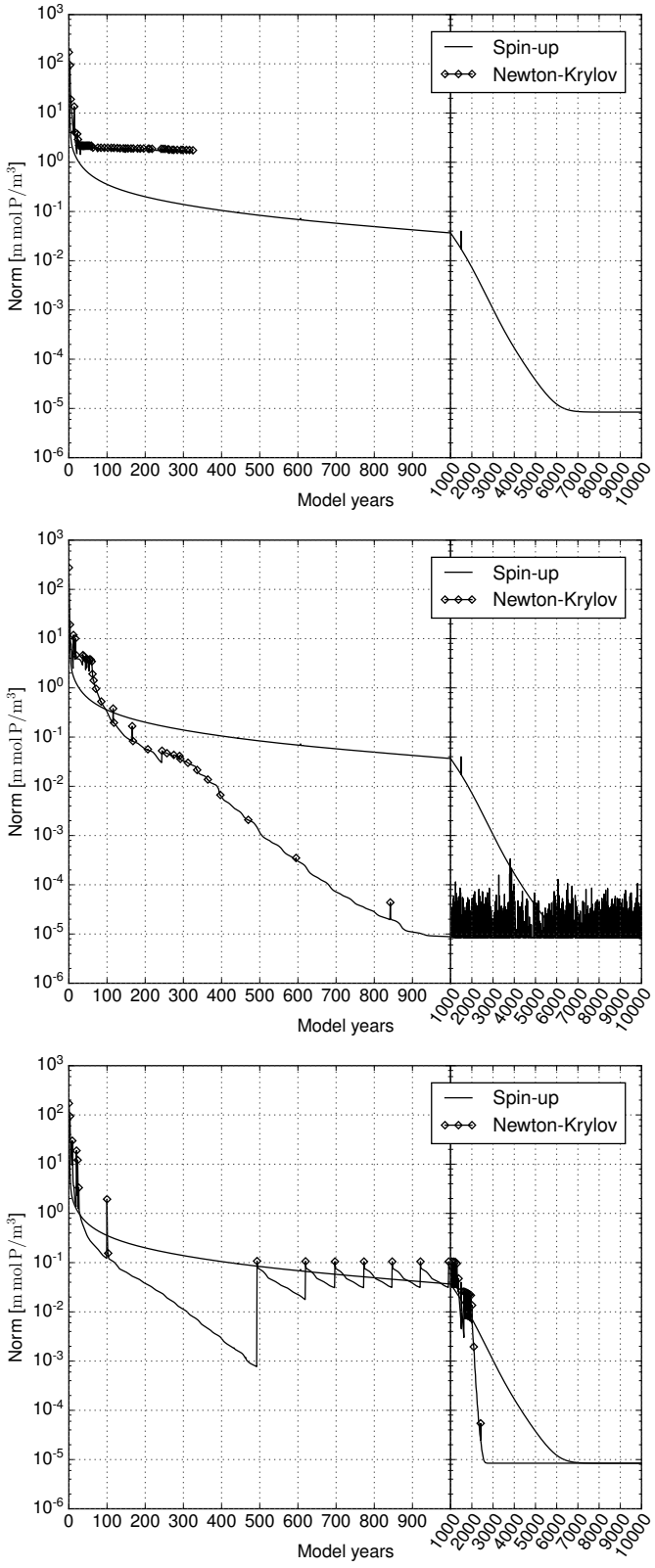

Figure 18. ~~Distribution of the computational time among main operations during the integration of~~ *NPZD-DOP model:* Convergence towards an annual cycle using a ~~model year~~spin-up and a Newton-Krylov solver. *Top:* Default Newton-Krylov setting. *Middle:* Changed initial value to $0.0434\ \mathrm{m\,mol\,P\,m^{-3}}$ for all tracers. *Bottom:* Changed inner accuracy to $\gamma = 0.3$.

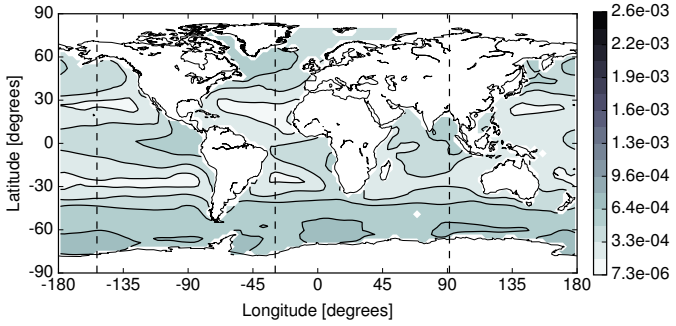

Figure 19. *MITgcm-PO4-DOP model:* Difference between the spin-up and Newton solution at the first layer (0 – 50 m) in the Euclidean norm.

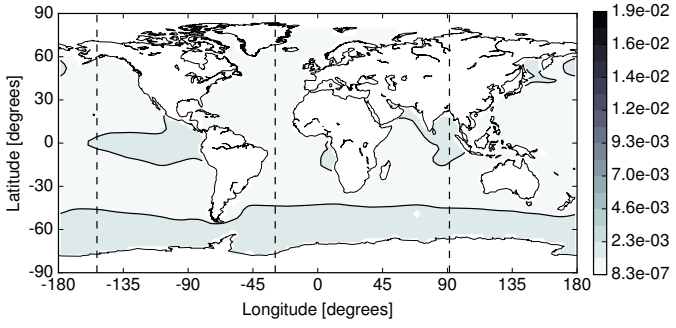

Figure 110. *N model:* Difference between the spin-up and Newton solution at the first layer (0 – 50 m) in the Euclidean norm.

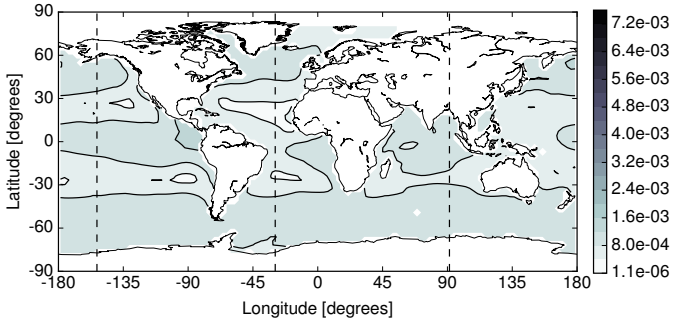

Figure 111. *N-DOP model:* Difference between the spin-up and Newton solution at the first layer (0 – 50 m) in the Euclidean norm.

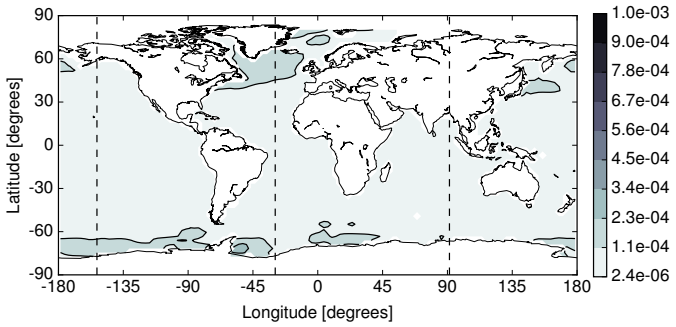

Figure 112. *NP-DOP model:* Difference between the spin-up and Newton solution at the first layer (0 – 50 m) in the Euclidean norm.

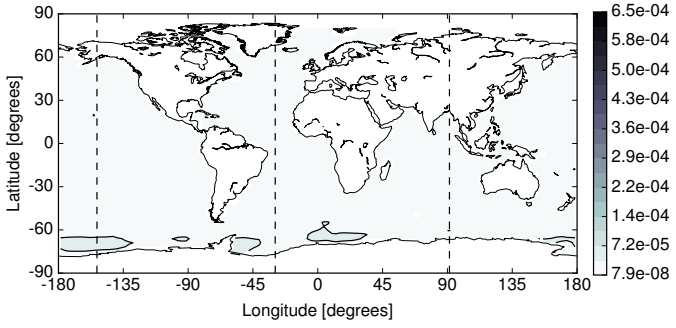

Figure 113. *NPZ-DOP model:* Difference between the spin-up and Newton solution at the first layer (0 – 50 m) in the Euclidean norm.

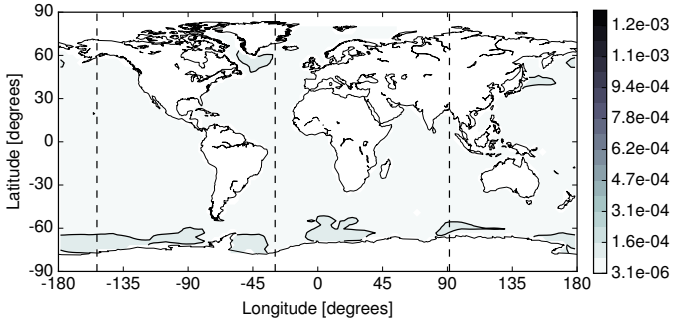

Figure 114. *NPZD-DOP model:* Difference between the spin-up and Newton solution at the first layer (0 – 50 m) in the Euclidean norm.

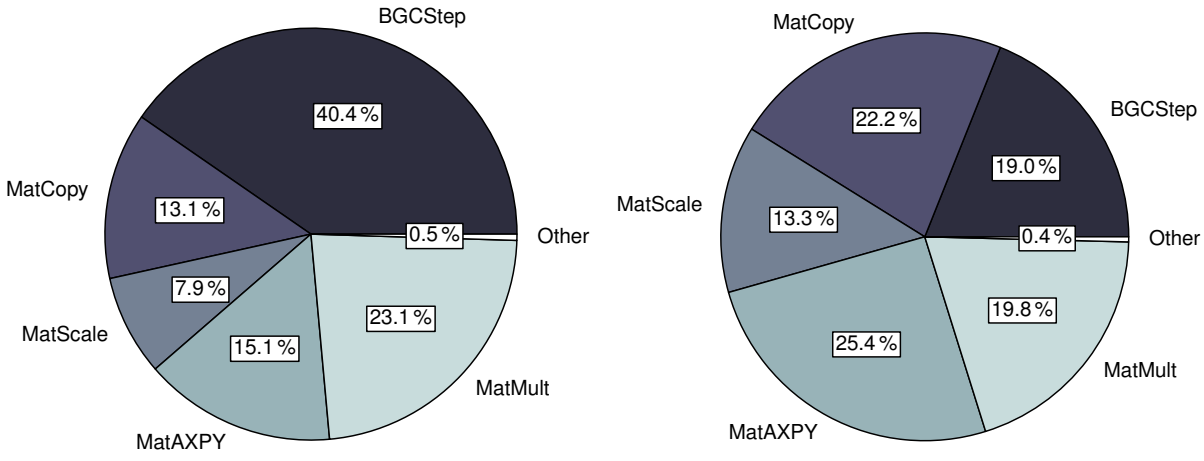

Figure 115. Distribution of the computational time among main operations during the integration of a model year. *Left:* MITgcm-PO4-DOP model. *Right:* N model.

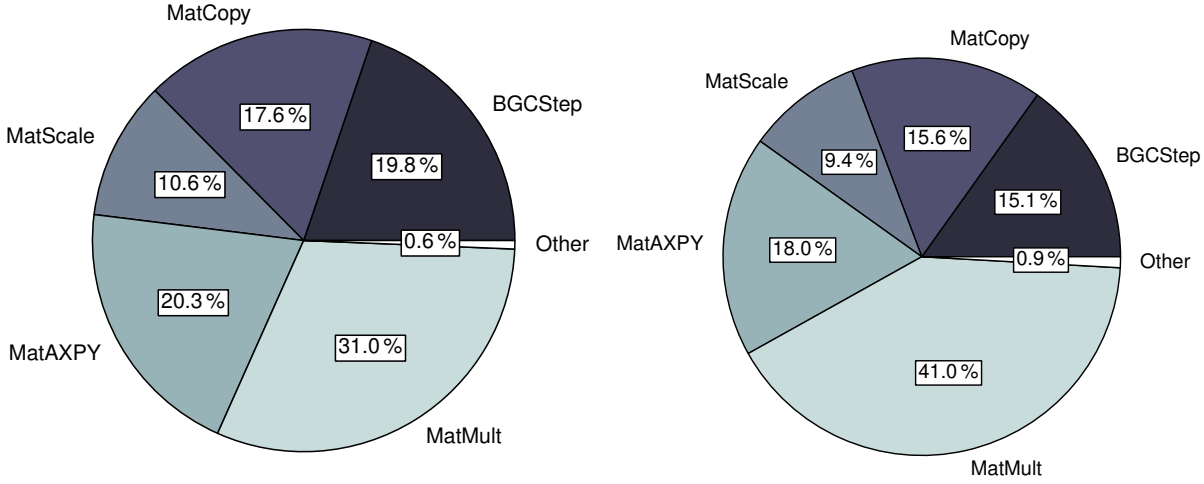

Figure 116. Distribution of the computational time among main operations during the integration of a model year. *Left:* N-DOP model. *Right:* NP-DOP model.

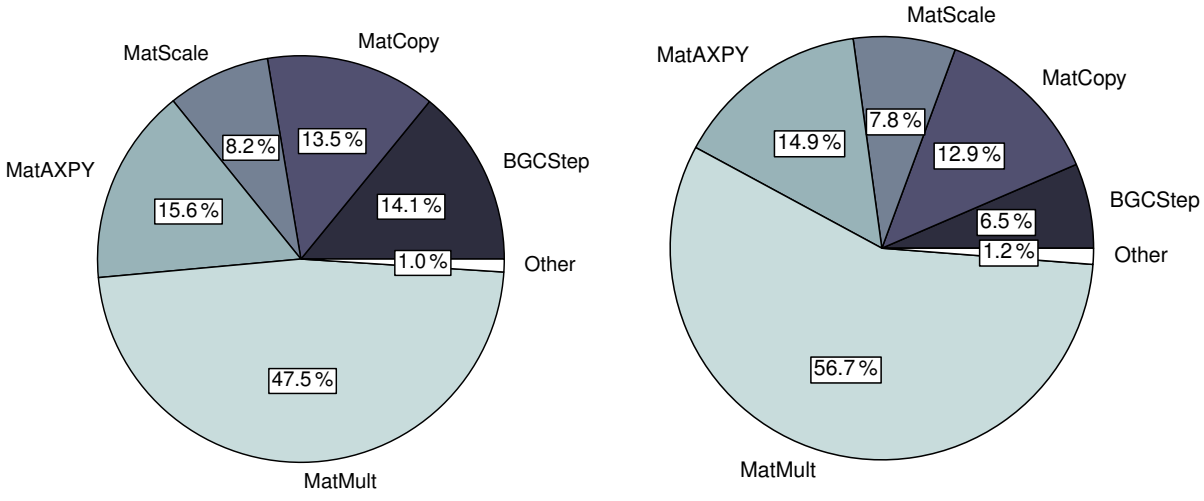

Figure 117. Distribution of the computational time among main operations during the integration of a model year. *Left:* NPZ-DOP model. *Right:* NPZD-DOP model.

Twin experiment using the spin-up solver. Bullets on the black line depict the steps of the optimization process. The gray line shows all model evaluations including gradient computation and line search step. The dashed line depicts a restart after 157 model evaluations. Vertical limits of the figures are also parameter bounds (except cost function and second parameter).

Twin experiment using the Newton solver. The black line depicts the steps taken by the optmizer using a absolute finite difference step that equals to the square root of the machine precision. The gray line refers to a relative finite difference step of 10^{-4}. Intermediate model evaluations are not shown here.

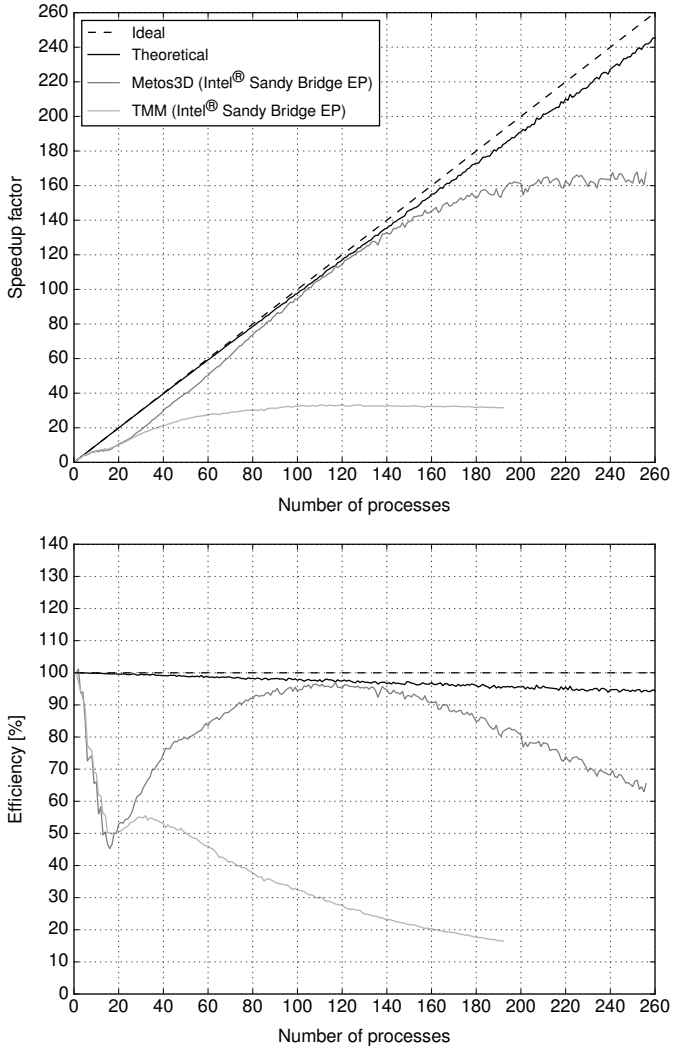

Figure 118. *MITgcm-PO4-DOP model:* Ideal and actual speedup factor as well as efficiency of parallelized computations. Here, ~~best possible~~ the notion theoretical refers to the used load distribution introduced in Section 7.4, i.e. a simulation run on an idealized hardware.

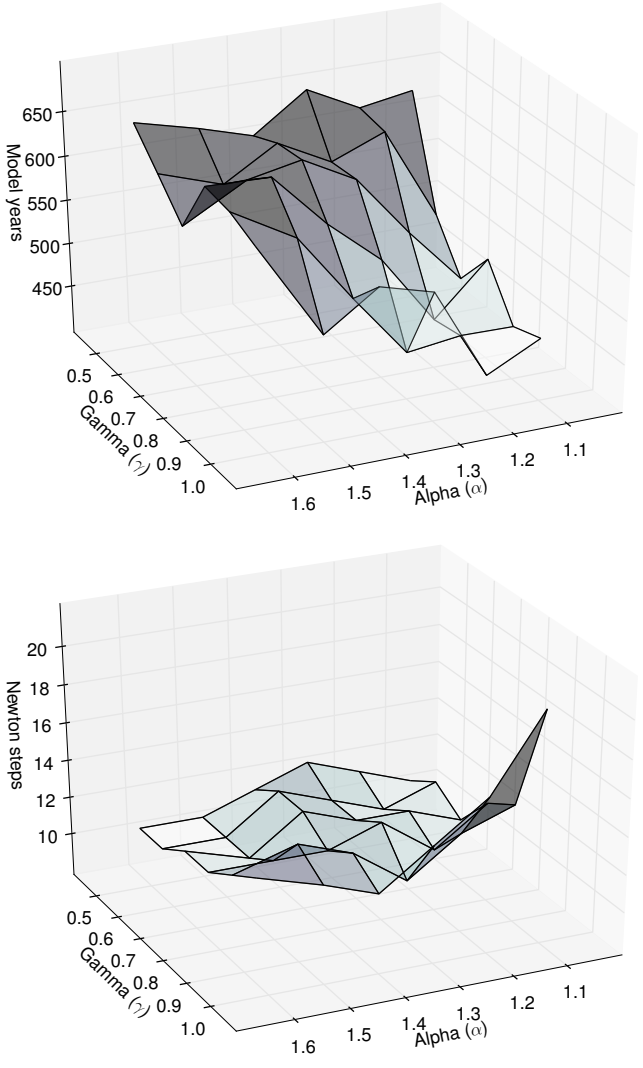

Figure 119. *MITgcm-PO4-DOP model:* Number of model years and Newton steps required for the computation of the annual cycle ~~$\mathbf{y}(\mathbf{u}_d)$~~ $\boldsymbol{y}(\boldsymbol{u}_d)$ as a function of different convergence control parameters α and γ (cf. Equation (10)).

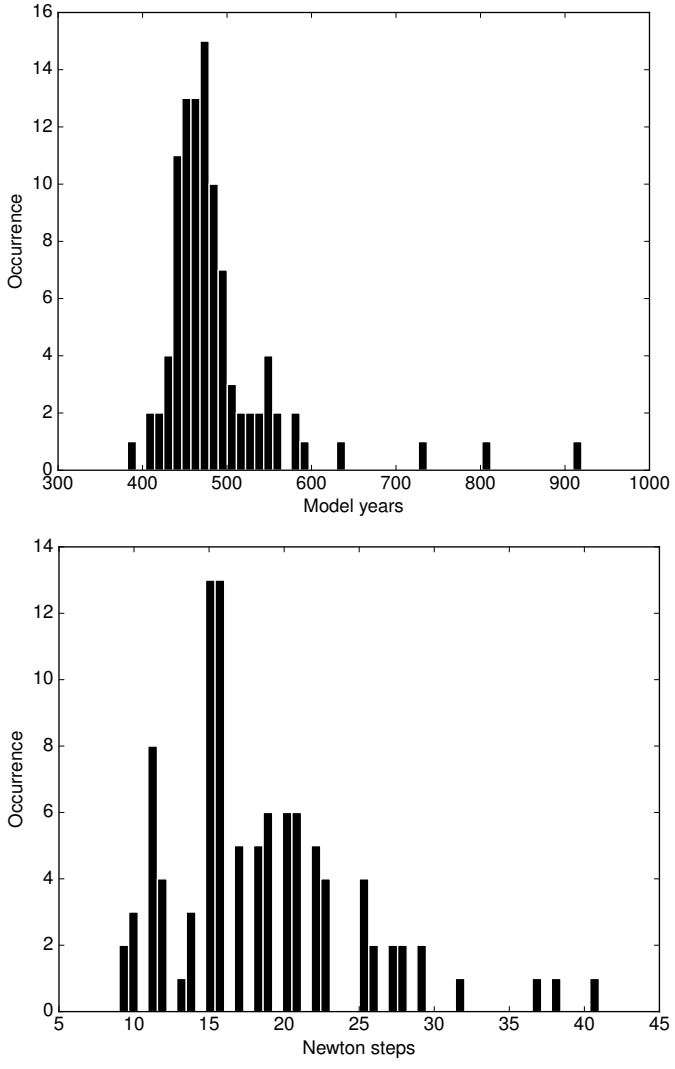

Figure 120. Distribution of number of model years and Newton steps required for the computation of a annual cycle using 100 random parameter samples (cf. Section 7.6).

Table 16. Difference in the Euclidean norm between the spin-up ($\boldsymbol{y}_S$) and the Newton ($\boldsymbol{y}_N$) solution. Regarding the NPZ-DOP and NPZD-DOP model a solution from the experiment with a different inner accuracy respectively a different initial value is used.

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 17. Vertical layers of the numerical model. Units are 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 18. 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	$1/\mathrm{y}$
u_2	α	2.0	1.5	200.0	$\mathrm{mmol P/m^3/y}$
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	$\mathrm{W/m^2}$
u_6	k	0.02	0.01	0.05	$1/\mathrm{m}$
u_7	a_{remin}	0.858	0.7	1.5	1

Table 19. 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	$(\mathrm{m\,mol\,P\,m}^{-3})^{-1}\mathrm{m}^{-1}$
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	$\mathrm{m\,mol\,P\,m}^{-3}$
K_P			0.088	0.088	0.088	$\mathrm{m\,mol\,P\,m}^{-3}$
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			$(\mathrm{m\,mol\,P\,m}^{-3})^{-1}\mathrm{d}^{-1}$
λ_Z				0.03	0.03	d^{-1}
κ_Z				3.2	3.2	$(\mathrm{m\,mol\,P\,m}^{-3})^{-1}\mathrm{d}^{-1}$
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	$\mathrm{d}^{-1}\mathrm{m}$

Algorithm 1: Load balancing

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

$w = 0$;
$n_{p,1\ldots N} = 0$;
for $k = 1, \ldots, n_p$ **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

Algorithm 2: Interpolation

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

$w = t * n_{data} + 0.5$;
$\beta = \text{mod}(w, 1.0)$;
$j_\beta = \text{mod}(\text{floor}(w), n_{data})$;
$\alpha = (1.0 - \beta)$;
$j_\alpha = \text{mod}(\text{floor}(w) + n_{data} - 1, n_{data})$;

Algorithm 3: `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: $\boldsymbol{y}_{out}$

~~$\boldsymbol{y}_{in} = \boldsymbol{y}_0$~~ $\boldsymbol{y}_{in} = \boldsymbol{y}_0$;
for $j = 1, \ldots, n_t$ **do**
    $t_j = \mod(t_0 + (j-1)\Delta t, 1.0)$;
    $\boldsymbol{y}_{out} = \texttt{PhiStep}(t_j, \Delta t, \mathbf{A}_{imp}, \mathbf{A}_{exp}, \boldsymbol{y}_{in}, \boldsymbol{u}, \boldsymbol{b}, \boldsymbol{d})$;
    $\boldsymbol{y}_{in} = \boldsymbol{y}_{out}$;
end

Algorithm 4: `PhiStep` (φ)

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: $\boldsymbol{y}_{out}$

~~$\mathbf{q} = \texttt{BGCStep}(t_j, \Delta t, \boldsymbol{y}_{in}, \boldsymbol{u}, \boldsymbol{b}, \boldsymbol{d})$~~ $\mathbf{q} = \texttt{BGCStep}(t_j, \Delta t, \boldsymbol{y}_{in}, \boldsymbol{u}, \boldsymbol{b}, \boldsymbol{d})$;
~~$\boldsymbol{y}_w = \texttt{TransportStep}(t_j, \mathbf{A}_{exp}, \boldsymbol{y}_{in})$~~ $\boldsymbol{y}_w = \texttt{TransportStep}(t_j, \mathbf{A}_{exp}, \boldsymbol{y}_{in})$;
~~$\boldsymbol{y}_w = \boldsymbol{y}_w + \mathbf{q}$~~ $\boldsymbol{y}_w = \boldsymbol{y}_w + \mathbf{q}$;
~~$\boldsymbol{y}_{out} = \texttt{TransportStep}(t_j, \mathbf{A}_{imp}, \boldsymbol{y}_w)$~~ $\boldsymbol{y}_{out} = \texttt{TransportStep}(t_j, \mathbf{A}_{imp}, \boldsymbol{y}_w)$;

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
```