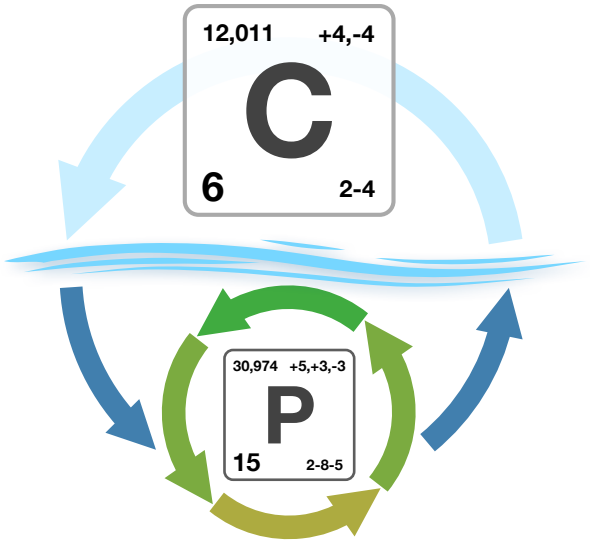
Parameter Estimates for Marine Ecosystem Models in 3-D



Dissertation

zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften
(Dr. rer. nat.)

der Technischen Fakultät
der Christian-Albrechts-Universität

vorgelegt von
Dipl.-Math. Jaroslaw Piwonski

Kiel, 2015

Referent:	Prof. Dr. Thomas Slawig
1. Koreferent:	Prof. Dr. Andreas Oschlies
2. Koreferent:	Prof. Dr. Steffen Börm
Tag der mündlichen Prüfung:	21. April 2015
Zum Druck genehmigt:	21. April 2015

There's probably no god.
Now stop worrying and enjoy your life.

Abstract

The aim of this work is to provide a computational-science-based foundation for the parameter identification of marine ecosystem models. For this purpose a general programming interface is introduced to enable a flexible coupling of marine ecosystems to fluid dynamics on source code level. This interface fits into the biogeochemical model structure as well as into an optimization context.

Moreover, a parallel simulation and solver software is implemented that combines the introduced interface with an efficient, transport-matrix-based simulation. The software is founded on a free and portable programming library. It is written from scratch, basically validated and exemplary used for a derivative-based optimization experiment.

Part of the software additionally provides a basis for the numerical experiments carried out subsequently. They address an approach used for the computation of sensitivities with respect to model parameters and an alternative optimization approach that does not require model evaluations, respectively.

In addition, results are included in this work that has been achieved in collaboration with other authors. The first joint work is about porting the software to graphic processing units, the second is about its usage for surrogate-based optimization.

Zusammenfassung

Das Ziel der vorliegenden Arbeit ist es, eine informationstechnische Grundlage für die Parameteridentifikation bei marinen Ökosystemmodellen zu schaffen. Dazu wird eine Programmierschnittstelle vorgestellt, die es allgemein ermöglicht, marine Ökosysteme auf Quelltextebene an Strömungsmodelle zu koppeln. Diese Schnittstelle fügt sich sowohl in die biogeochemische Modellstruktur als auch in den Optimierungskontext ein.

Des weiteren wird eine parallele Simulations- und Lösungssoftware implementiert, die eine effiziente, transportmatrixbasierte Strömungssimulation und die vorgestellte Schnittstelle miteinander kombiniert. Die Software wird auf der Grundlage einer freien und portablen Bibliothek neu erstellt, grundsätzlich validiert und exemplarisch für ein ableitungsbasiertes Optimierungsverfahren eingesetzt.

Ein Teil der Software bildet zusätzlich eine Basis für die im weiteren Verlauf der Arbeit durchgeführten numerischen Experimente. Dabei handelt es sich um einen Ansatz zur Berechnung von Sensitivitäten bezüglich der Modellparameter, beziehungsweise um einen alternativen Optimierungsansatz, der ohne Modellauswertung auskommt.

Zusätzlich werden Resultate in die Arbeit aufgenommen, die in Zusammenarbeit mit anderen Autoren erzielt wurden. Es handelt sich dabei um die Portierung der Software auf Grafikkarten und um deren Einsatz im Bereich der surrogat-basierten Optimierung.

CONTENTS

INTRODUCTION

The earth's climate is a complex and subtly tuned system of interlocking processes on many spatial and temporal scales. It consists of three major components: land (soils and plants), atmosphere and ocean. Among and within these components exist matter fluxes based on physical, chemical and biological processes. In the context of climate change the global carbon cycle, including carbon dioxide (CO_2), is of particular interest. For details refer to Fasham (2003, F03 hereafter, pp. 157), Sarmiento and Gruber (2006, SG06 hereafter, pp. 392) or Stocker et al. (2013, IPCC13 hereafter, pp. 465).

According to estimates, the sizes of the pre-industrial reservoirs of carbon has been 2,000 Gt C[1], 590 Gt C and 38,000 Gt C regarding land, atmosphere and ocean, respectively (F03, p. 124). In the last 250 years, by combustion of fossil fuels, deforestation and extensive land use, man has significantly interfered with the global carbon cycle. From 1750 to 2011 on average 555 Gt C were released into the atmosphere, of which 160 Gt C accumulated in natural terrestrial ecosystems, 240 Gt C remained in the atmosphere and 155 Gt C has been taken up by the oceans (IPCC13, p. 12).

Remarkably, the uptake ratio does not reflect the relation between the pre-industrial reservoirs at all. In this regard, the work of many researches in the field of ocean biogeochemistry have contributed during the last decades to understand this specific distribution. The effort revealed a sophisticated balance between biological and physical processes in the oceanic carbon cycle. A concept of the so-called biological pump(s) and gas exchange pump became familiar (cf. SG06, p. 342).

[1] Gt C: giga (billion) tons of carbon

However, "(...) while the biological pump does not play a significant role for assessing the oceanic uptake of anthropogenic CO2 in the past and present, these pumps might play an important role for the future uptake." (cf. SG06, p. 417) For instance, the ocean biota gradually begins to react to the acidification of marine waters, which is caused by the rising CO_2 concentrations. Hence, it becomes more and more important to explicitly take the ecosystem into account.

In this regard, marine ecosystem models still entail many uncertainties concerning the number of components and parameterizations. "In particular, there is no general consensus on how complex a biogeochemical model should be in order to faithfully represent the interplay between ocean biota, ocean physics, and the marine biogeochemical cycles of carbon and the major nutrients." (cf. Kriest et al., 2010)

A wide range of models needs to be validated and assessed regarding their ability to reproduce the real world system. This involves a professional discussion of simulation results and, preferably, an estimation of optimal model parameters beforehand. (cf. Fennel et al., 2001; Schartau and Oschlies, 2003). 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 too high, even at lower resolution, considering optimization methods that may require hundreds of model evaluations. 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).

In this context, the aim of this work is to introduce a general programming interface for parameter identification for biogeochemical and marine ecosystem models. Moreover, a comprehensive solver software for periodic steady-states is implemented that includes a fixed point iteration (spin-up) and a Newton solver (cf. Piwonski and Slawig, 2015, Metos3D). The software is based on the Portable, Extensible Toolkit for Scientific Computation (Balay et al., 2012, PETSc) library and uses transport matrices for efficient simulation in 3-D.

In the following sections, we briefly recapitulate the theoretical background of marine ecosystem dynamics, the used techniques to solve the underlying equations and parameter identification. In this regard, the most

text passages are taken literally from the Metos3D paper included in the appendix.

1.1 Marine ecosystem dynamics

We consider the following off-line tracer transport model, which is described by a system of nonlinear time-dependent partial differential equations (PDEs) defined on the unit time interval $I = [0, 1[\subset \mathbb{R}$ (a model year), a spatial domain $\Omega \subset \mathbb{R}^3$ and its boundary $\Gamma = \partial\Omega$. For n tracers the system generally reads

$$\frac{\partial y_i}{\partial t} = \nabla \cdot (\kappa \nabla y_i) - \nabla \cdot (v\, y_i) + q_i(y, \mathbf{u}, b, d), \tag{1.1}$$

where y_i is a tracer concentration with $y_i : I \times \Omega \to \mathbb{R}$ and $y = (y_i)_{i=1}^n$ is a vector of all tracers. Additionally, homogeneous Neumann boundary conditions on the entire Γ for all tracers y_i are imposed. 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.

The transport of tracers in marine waters is depicted by a diffusion and an advection term. The diffusion mixing coefficient $\kappa : I \times \Omega \to \mathbb{R}$ and the advection velocity field $v : I \times \Omega \to \mathbb{R}^3$ are regarded as given (cf. next Section). Note that both operators effect each tracer separately. In contrast, a single component of the biogeochemical model q_i may generally depend on all tracers, i.e.

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

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 insolation or wind speed 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 salinity or temperature of the ocean water. The model also includes parameters which are summarized in the vector $\mathbf{u} \in \mathbb{R}^m$ and kept temporally as well as spatially constant during the computation of a model year.

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. Accordingly, we solve the model equations by computing an annual cycle, which is a vector of tracer concentrations with $y(t + 1, x) = y(t, x)$ (cf. Section 1.2).

Transport matrices

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

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

where $L(t)$ comprises both and represents a time dependent linear operator. Formally, its fully discrete equivalent is a sequence of matrices $(\mathbf{L}_j)_{j=1}^{n_t}$ with $\mathbf{L}_j = \mathbf{L}(t_j)$. 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 $\Delta t = 1/n_t$.

However, the matrices that we use throughout this work represent the effect of an entire time step. 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, p. 267).

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

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

Here, the transport is split as $\mathbf{L}_j = \mathbf{L}_{imp,j} + \mathbf{L}_{exp,j}$ and $\mathbf{I}$ represents the identity. Throughout this work, both matrix types are sparse. The implicit matrix $\mathbf{L}_{imp,j}$ comprises vertical diffusion only, i.e. a process within a water column that is computed and inverted independently of its vicinity. The explicit matrix $\mathbf{L}_{exp,j}$ represents a (local) differential operator, which naturally has a sparse discrete representation.

Overall, the fully discrete iteration scheme for n tracers results in a block diagonal system. The integration of state variables over a model year consists of sparse matrix vector multiplications and evaluations of the biogeochemical model. For a fixed time index j it reads

$$\begin{aligned}\mathbf{y}_{j+1} &= \mathbf{A}'_{imp,j}(\mathbf{A}'_{exp,j}\, \mathbf{y}_j + \Delta t\, \mathbf{q}_j(\mathbf{y}_j, \mathbf{u}, \mathbf{b}_j, \mathbf{d}_j)) \\ &= \varphi_j(\mathbf{y}_j, \mathbf{u}, \mathbf{b}_j, \mathbf{d}_j)\,,\end{aligned} \tag{1.2}$$

Listing 1.1: Fortran 95 implementation of the coupling interface for biogeochemical and marine ecosystem models.

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

where $\mathbf{y}_j = (\mathbf{y}_i(t_j))_{i=1}^n$ combines all discrete tracer vectors. 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.

Biogeochemistry

For the time being, we assume that biogeochemical and marine ecosystem models are implemented for a single water column only, synonymously called profile in the following. This reflects the fact that the most important non-local biogeochemical processes happen therein (cf. Evans and Garçon, 1997). Thus, 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 evaluation of the whole n 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

$$\Delta t \left(\mathbf{q}_i(t_j, (\mathbf{y}_i)_{i=1}^n, \mathbf{u}, (\mathbf{b}_i)_{i=1}^{n_b}, (\mathbf{d}_i)_{i=1}^{n_d})\right)_{i=1}^n . \tag{1.3}$$

In this regard, Listing 1.1 shows a realization of the biogeochemical model interface in Fortran 95 that was introduced by Piwonski and Slawig (2015).

1.2 Periodic solutions

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 (1.1) 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

$$\phi(\mathbf{y}_1, \mathbf{u}) = \mathbf{y}_1, \tag{1.4}$$

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 (1.2)). 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.

Assuming there exist a unique solution of Equation (1.1), 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 (normalized) counterpart by

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

where w_k is the *relative* volume of the partial grid box Ω_k, assuming the domain is scaled to a unit cube. 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 employ an unweighted norm that only compares the initial states of consecutive model years. For a fixed time index j we then denote

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

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. Here, the differ-

ence between consecutive iterates is determined as

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

where $l = 2, \ldots, n_l$.

Inexact Jacobian-free Newton-Krylov

Alternatively, Equation (1.4) 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

$$F'(\mathbf{y}_k)\, \mathbf{s}_k = -F(\mathbf{y}_k)\,, \tag{1.5}$$

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. $\mathbf{y}_{k+1} = \mathbf{y}_k + \mathbf{s}_k$. To solve the system of linear equations for a fixed k we use a Krylov subspace approach. These solvers require only the *result* of a matrix-vector product to proceed. 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 $\mathbf{s}$.

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

$$\eta_k = \gamma \left(\frac{\|F(\mathbf{y}_k)\|_2}{\|F(\mathbf{y}_{k-1})\|_2} \right)^{\alpha} \tag{1.6}$$

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

1.3 Parameter identification

In this context, parameter identification for biogeochemical and marine ecosystem models is regarded as a so-called PDE-constrained optimization problem. It is generally denoted as

$$\min_{\mathbf{u} \in U} J(y, \mathbf{u}) \quad s.t. \quad e(y, \mathbf{u}) = 0\,,$$

where y is the state and $\mathbf{u}$ the control or design variable. The former is constrained to the system of model equations, which is defined by Equation (1.1) and represented by $e(y, \mathbf{u}) = 0$. It is assumed to be uniquely defined thereby. The latter is constrained to an admissible set U, which is the m-dimensional hyper rectangle defined by the bound constraints for the model parameters (see below). Moreover, J is usually referred to as objective, often as cost or misfit function. For details refer to Herzog (2010) and Borzì and Schulz (2012, pp. 3).

The problem is tackled by either an all-at-once or a black-box method. Regarding the former, $e(y, \mathbf{u}) = 0$ is explicitly kept as a side constraint during the optimization. This approach implies the formulation of a Lagrangian and consequently involves an adjoint model (Herzog, 2010). However, considering a long primal iteration, i.e. a spin-up for instance, the computation of the dual state variable requires the entire (primal) trajectory.

Here, a naive approach, at which the whole trajectory is stored, would produce 3.31 TB data per tracer, regarding a $2.8125°$ resolution, a spin-up of 3,000 model years and 2,880 time steps per model year. Consequently, special techniques are required, one of which is the so-called one-shot method (cf. Griewank and Hamdi, 2011, 2010). In this regard, based on the ideas of Metos3D, a software was successfully implemented in collaboration with Claudia Kratzenstein.

Howsoever, only black-box methods were considered in the context of this work. This implies that the PDE constraints are eliminated by means of a solution operator $y = S(\mathbf{u})$ (cf. Herzog, 2010). Consequently, for the identification of optimal model parameters, we consider the following, so-called reduced, optimization problem:

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

where

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

and the admissible set is defined as

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

Here, $\mathbf{y}(\mathbf{u})$ is the model output, i.e. a solution of the model equations for a given parameter set $\mathbf{u}$, and $\mathbf{y}_d$ is the (observational) data to which it is compared. Moreover, $\mathbf{b}_l$ respectively $\mathbf{b}_u$ are the lower and upper bounds we impose during the optimization (Nocedal and Wright, 2000, pp. 304).

All papers listed in the Appendix should be understood in this context. In (Piwonski and Slawig, 2015) an information-science-based foundation for the evaluation of marine ecosystem models for such black-box methods is presented. An efficient parallel software is implemented and exemplarily used for a derivative-based black-box optimization algorithm. Moreover, the acceleration of model evaluations is the subject of (Siewertsen et al., 2013). Here, the software was successfully ported to a graphic processing unit (GPU). Furthermore, Metos3D has also been used for surrogate-based optimization (Prieß et al., 2012, 2013). In this regard, a MATLAB interface for coarse and fine model evaluations has been implemented.

PRESENT RESEARCH

This chapter presents the ongoing research that is founded on the basic studies presented in (Piwonski and Slawig, 2015). The following results were chosen to demonstrate a general applicability of the introduced programming interface and how the implemented building blocks of Metos3D can be used for other approaches. However, the presented surveys are understood as feasibility studies. Thus the introductions are kept brief and the used software is not freely available.

First of all, we carry out numerical experiments with an NPZD-DOP model as a representative of marine ecosystem models. Next, we discuss how parameter sensitivities for a converged periodic steady-state can be computed at little additional computational cost. In this regard, a similar approach was taken by Kwon and Primeau (2006) for equilibrium solutions, whereas a resolved seasonal cycle is used here. At last, an alternative optimization approach is considered, at which no expensive model evaluations are required. Referring to this, an introduction is provided by (cf. Banks and Kunisch, 1989, pp. 55) for instance.

2.1 The NPZD-DOP model

Since we used only a biogeochemical model for the basic validation in (Piwonski and Slawig, 2015), results from numerical experiments with a marine ecosystem model are presented in this section to continue the validation. The source code of the NPZD-DOP model was kindly provided by Iris Kriest. It has been described and used for investigations in Kriest et al. (2010).

The model comprises six biogeochemical variables, namely nutrients (N),

phytoplankton (P), zooplankton (Z), detritus (D), dissolved organic phosphorous (DOP) and additionally oxygen (O2), which is not mentioned in the section title. The introduced parameters are shown in Table 2.1. As described in (Piwonski and Slawig, 2015) we implement a wrapper for the provided source code, store all files in a appropriate folder and compile all sources to create the final executable. Note that the order in which the tracers are implemented within the model is different as mentioned above. Internally, the tracer vector is $\mathbf{y} = (\mathbf{y}_{\mathrm{N}}, \mathbf{y}_{\mathrm{DOP}}, \mathbf{y}_{\mathrm{O2}}, \mathbf{y}_{\mathrm{P}}, \mathbf{y}_{\mathrm{Z}}, \mathbf{y}_{\mathrm{D}})$. This is just a technical detail, but it is important for the preparation of the options files.

Thus prepared, we carry out two solver experiments. First, we let the spin-up iterate for 10,000 model years with no set tolerance. Next, we let the Newton solver compute an annual steady-state of the NPZD-DOP model. Regarding the latter, we leave all settings to default, except for the maximum number of Newton steps, which is set to 150. In both experiments, the tracers are initialized as depicted in Table 2.2.

Figure 2.1 shows the convergence towards a periodic steady-state using the spin-up respectively Newton solver. We observe that the spin-up requires almost 10,000 model years to converge, whereas the Newton solver reaches the same tolerance in less than 1,500 model years. This is at least 6 times faster and confirms the results from (Piwonski and Slawig, 2015) for the MITgcm-PO4-DOP model. Moreover, the computed concentration of N (cf. Figure 2.2) resembles the one shown by (Kriest et al., 2010). We regard this as an additional proof that the implemented software is working correctly and the programming interface is capable of coupling a variety of biogeochemical and marine ecosystem models. For the sake of completeness, slices through the Pacific, the Atlantic and the Indian are shown in Figure 2.3.

2.2 Model sensitivity

The model sensitivity, or more precisely, the sensitivity of the model output with respect to the introduced parameters, is a common means for assessment of biogeochemical models (cf. Kwon and Primeau, 2006). In this section we present an approach for the computation of such model sensitivities at little computational cost once a solution, i.e. periodic steady-state, has already been found.

Let us again assume ϕ is the mapping that integrates given tracer concen-

Table 2.1: Parameters implemented in the NPZD-DOP model. Specified are the location within the parameter vector, the variable name within the implementation, the value used for the computed solution and the symbol of each parameter used in (Kriest et al., 2010). The constant rnp is the Redfield ratio of N and P, i.e. $\mathrm{rnp} = 16$.

u	Variable name	Value	Symbol
u_1	`ACmuphy`	0.6	μ_{PHY}
u_2	`ACik`	24.0	I_c
u_3	`ACkpo4`	0.5 / rnp	K_{PHY}
u_4	`ACkchl`	0.03 * rnp	k_c
u_5	`ACkw`	0.04	k_w
u_6	`AClambda`	0.03	λ_{PHY}
u_7	`plambda`	0.01	λ'_{PHY}
u_8	`ACmuzoo`	2.0	μ_{ZOO}
u_9	`AClambdaz`	0.03	λ_{ZOO}
u_{10}	`AComniz`	0.2 * rnp	κ_{ZOO}
u_{11}	`ACeff`	0.75	ϵ_{ZOO}
u_{12}	`zlambda`	0.01	λ'_{ZOO}
u_{13}	`graztodop`	0.15	σ
u_{14}	`dlambda`	0.17 / 360.0	λ'_{DOP}
u_{15}	`detlambda`	0.05	λ'_{DET}

Table 2.2: Initial tracer concentrations used for the computation of periodic steady-states of the NPZD-DOP model. The values represent global means.

Tracer	Value	Unit
N	2.17	$\mathrm{mmol\,P/m^3}$
DOP	0.0001	$\mathrm{mmol\,P/m^3}$
O2	176.6	$\mathrm{mmol\,O_2/m^3}$
P	0.0001	$\mathrm{mmol\,P/m^3}$
Z	0.0001	$\mathrm{mmol\,P/m^3}$
D	0.0001	$\mathrm{mmol\,P/m^3}$

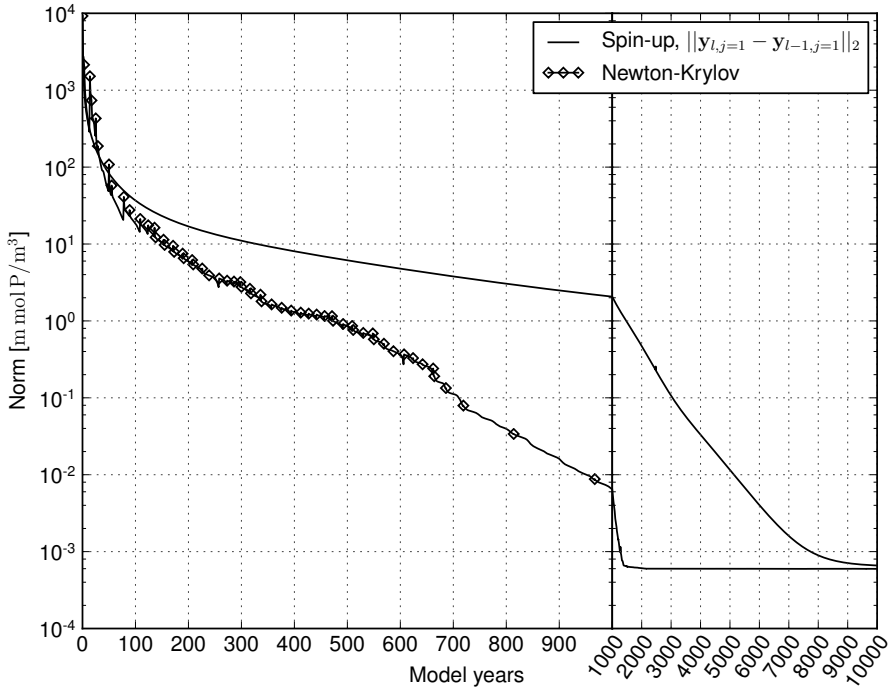


Figure 2.1: Convergence towards an annual cycle of the NPZD-DOP model. *Spin-up:* norm of difference between initial states of consecutive model years (solid line). *Newton-Krylov:* residual norm at a Newton step (diamond) and norm of the GMRES residual during solving (solid line in-between).

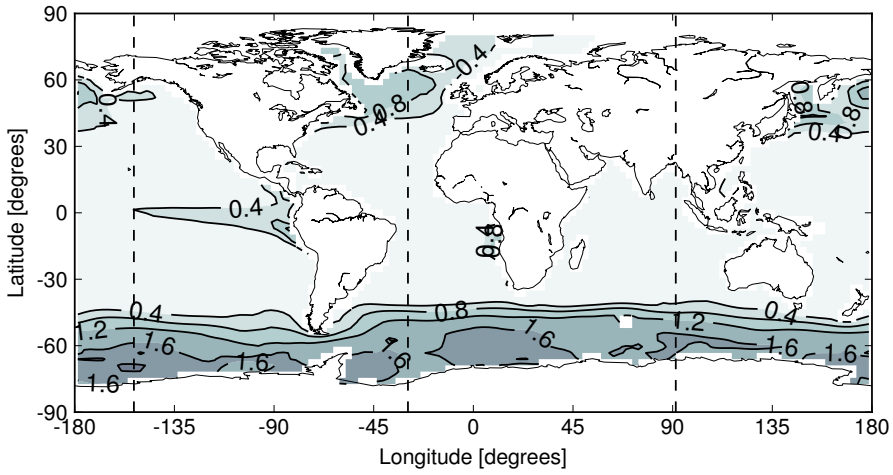


Figure 2.2: Concentration of phosphate ($\mathbf{y}_\mathrm{N}$) at the first layer (0 – 50 m). Shown is the initial state (1st of January, 00:00 am) of the converged annual cycle presented in Figure 2.1. The dashed lines depict locations of slices shown in Figure 2.3.

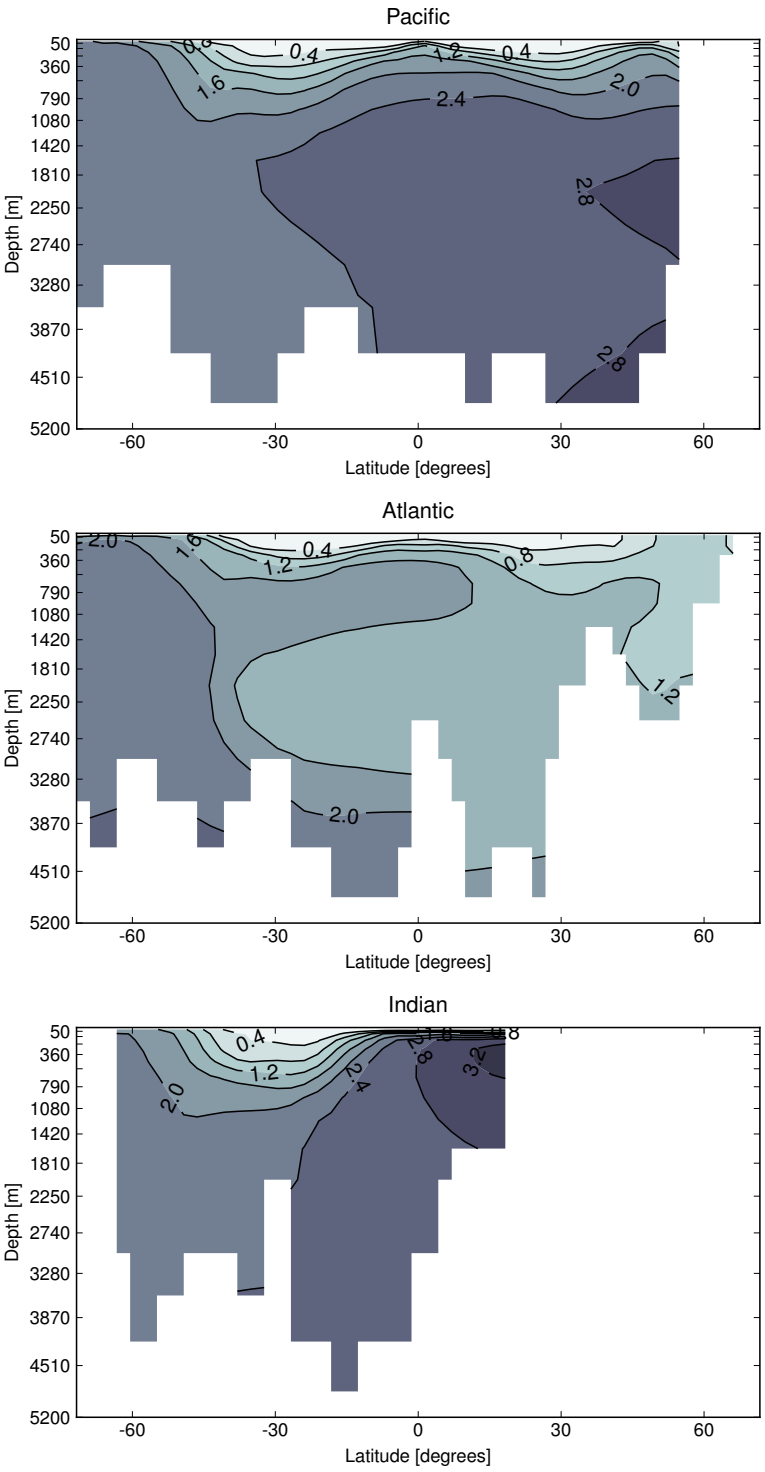


Figure 2.3: Slices corresponding to Figure 2.2: the Pacific (153.2815° W), the Atlantic (29.53125° W) and the Indian (91.40625° E).

trations over a model year. And let y be an annual steady-state with respect to a given parameter set $u \in \mathbb{R}^m$, i.e. a fixed point of ϕ. Hence,

$$y(u) = \phi(y(u), u)\,,$$

where the dependency on u is explicitly noted here. Then the derivative with respect to u formally yields:

$$\begin{aligned} \frac{d}{du}y(u) &= \frac{d}{dy}\phi(y,u)\frac{d}{du}y(u) + \frac{d}{du}\phi(y,u) \\ (I - \frac{d}{dy}\phi(y,u))\frac{d}{du}y(u) &= \frac{d}{du}\phi(y,u)\,. \end{aligned} \tag{2.1}$$

Let us now introduce $F(y,u) = y - \phi(y,u)$ and rewrite Equation (2.1) as

$$F'(y,u)\,\nabla_u\, y(u) = \nabla_u\, \phi(y,u)\,, \tag{2.2}$$

where F' is the Jacobian of F with respect to y and $\nabla_u = (\partial_{u_1}, \ldots, \partial_{u_m})$ is the gradient with respect to the model parameters.

Here, the columns of $\nabla_u\, y(u)$ and $\nabla_u\, \phi(y,u)$ are the desired sensitivities respectively the corresponding right hand sides. We recognize that for each column Equation (2.2) represents a similar system of linear equations as the one depicted in Equation (1.5). Thus, to solve Equation (2.2) we just require the right hand side, since the Jacobian is exactly the same that is used in each Newton step. Consequently, we reuse the already implemented F and the linear equation solver that is provided by PETSc.

We decide to compute a sensitivity with respect to the first parameter of the MITgcm-PO4-DOP model, i.e. $\partial_{u_1} y(u)$, which is the DOP remineralization rate (cf. Table 2; Piwonski and Slawig, 2015). Assuming the fixed point is already given, we approximate the corresponding right hand side, i.e. $\partial_{u_1}\phi(y,u)$, by finite differences, which requires the evaluation of *one* model year only. We set up the Krylov sub-space solver exactly as for the Newton approach, except we set the tolerance to 10^{-3}. The solver reaches the tolerance after 210 model years. Thus, overall, the computation of the sensitivity required the simulation of additional 221 model years.

Figure 2.4 and Figure 2.5 depict the computed PO4 and DOP sensitivities, respectively. As expected, the former is positive and the latter negative, which corresponds to the signs within the model equations (cf. Dutkiewicz et al., 2005). We regard this as an indication of a correct implementation. However, the figures do not resemble the sensitivities presented by Kwon

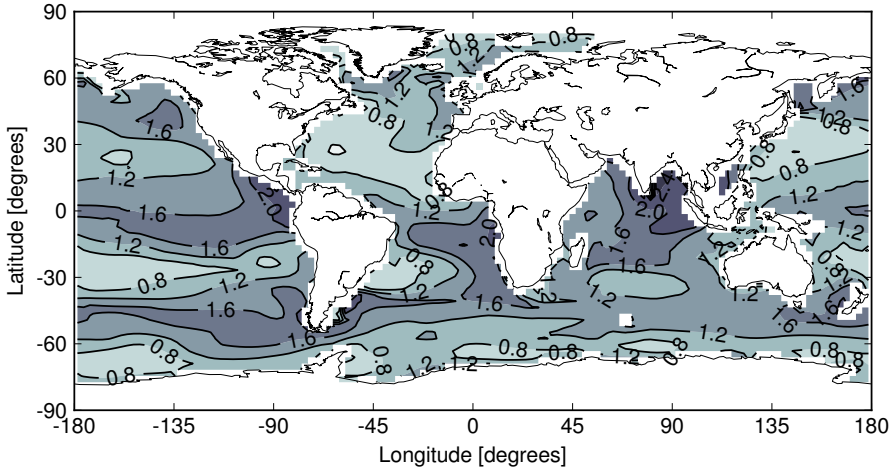


Figure 2.4: Column averaged PO4 sensitivity with respect to the DOP remineralization rate of the initial state (1st of January, 00:00 am) of the converged annual cycle. The unit is $\mathrm{mmol\,m^{-3}\,y^{-1}}$.

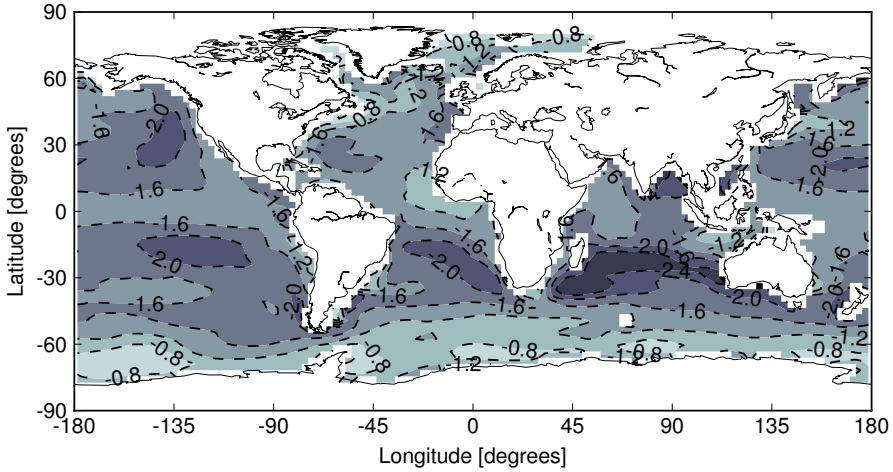


Figure 2.5: Column averaged DOP sensitivity with respect to the DOP remineralization rate of the initial state (1st of January, 00:00 am) of the converged annual cycle. The unit is $\mathrm{mmol\,m^{-3}\,y^{-1}}$.

and Primeau (2006). Here, an additional investigation would be necessary to find out, if the observed differences are due to the data-driven model they use or the fact that a seasonal cycle was used here.

2.3 Equation error optimization

In this section we present results from an alternative approach for the optimization problem. Whereas the usual thinking is to evaluate a given model first and then compare the output to the observed data (output error). Here, the idea is to insert the given data directly into the equations and assess the error it generates (equation error).

Formally, we still consider the same optimization problem as described above. However, we redefine J as

$$\frac{1}{2}||\mathbf{y}(\mathbf{u}) - \mathbf{y}_d||^2_{2,I\times\Omega} \quad \rightarrow \quad \frac{1}{2}||\phi(\mathbf{y}_d, \mathbf{u}) - \mathbf{y}_d||^2_{2,I\times\Omega}$$

and ϕ as

$$\phi = \varphi_{n_t} \circ \cdots \circ \varphi_1 \quad \rightarrow \quad \phi = \begin{pmatrix} 0 & \dots & 0 & \varphi_{n_t} \\ \varphi_1 & & & 0 \\ & \ddots & & \vdots \\ & & \varphi_{n_t-1} & 0 \end{pmatrix}.$$

Hence, we dispose of the expensive model evaluation $\mathbf{y}(\mathbf{u})$ for a given parameter set $\mathbf{u}$ within the cost function. Moreover, we do not compose a trajectory from an initial state anymore. Instead, we apply each time step separately on the components of a given data trajectory. Subsequent, we compare the result to the corresponding state (of the given data trajectory), for instance $\mathbf{y}_1$ is compared to $\varphi_{n_t}(\mathbf{y}_{n_t})$, $\mathbf{y}_2$ to $\varphi_1(\mathbf{y}_1)$ and so on. Consequently, an evaluation of the cost function corresponds to the simulation of *one* model year only. Thus prepared, we can start to adjust the model equation parameters to find a minimum in the same way as for the output error.

In this context, we decide to use the whole potential of the underlying parallel simulation and implement an optimization software that is based on the Toolkit for Advanced Optimization (TAO; Munson et al., 2012), which since version 3.5 is part of the Portable, Extensible Toolkit for Scientific Computation (PETSc; Balay et al., 1997). For this purpose, we use the already implemented times step routine φ from Metos3D and chose a Quasi-Newton

approach with gradient projection, regarding the parameter constraints, as optimization algorithm (Munson et al., 2012, pp. 24).

We carry out 100 twin experiments for the MITgcm-PO4-DOP model using the 100 Latin Hypercube samples as initial parameters sets $\mathbf{u}_0$ and the reference solution $\mathbf{y}_d = \mathbf{y}(\mathbf{u}_d)$ described in (Piwonski and Slawig, 2015). We use 128 Intel Xeon E5-4640 processors running at 2.4 GHz and limit each optimization to 1,000 model years. This amounts to overall 66 hours of computational time.

Figure 2.6 depicts the decay of the cost function. We observe that most of the optimizations required between 700 and 900 model years. These computations stopped because the absolute function value tolerance was reached, which by default is set to 10^{-4}. Here, the parameters were identified with a *relative* accuracy of at least 10^{-3}. Overall, all twin experiments were successful as shown in Figure 2.7.

Unfortunately, as previously implied, the *whole* trajectory for *all* tracers is required for this approach. This seems not realistic in case of observational data, not even for the simple PO4-DOP model. Whereas enough measurements for phosphate (PO4) may be available, regarding dissolved organic phosphorous (DOP), only a few dispersed data points exist. The situation seems even more desperate, considering more complex models.

Here, a way out is to optimize the missing tracer data along with the model parameters. However, the required gradients cannot be approximated with finite differences anymore. Regarding a $2.8125°$ resolution (cf. Piwonski and Slawig, 2015), the trajectory of *one* tracer consists of 151,917,120 entries. Thus, an alternative approach must be considered. Here, it is recommended to use the so-called reverse mode known from the theory of Automatic Differentiation (cf. Griewank and Walther, 2008, pp. 45). However, at the present time the corresponding software is still under development.

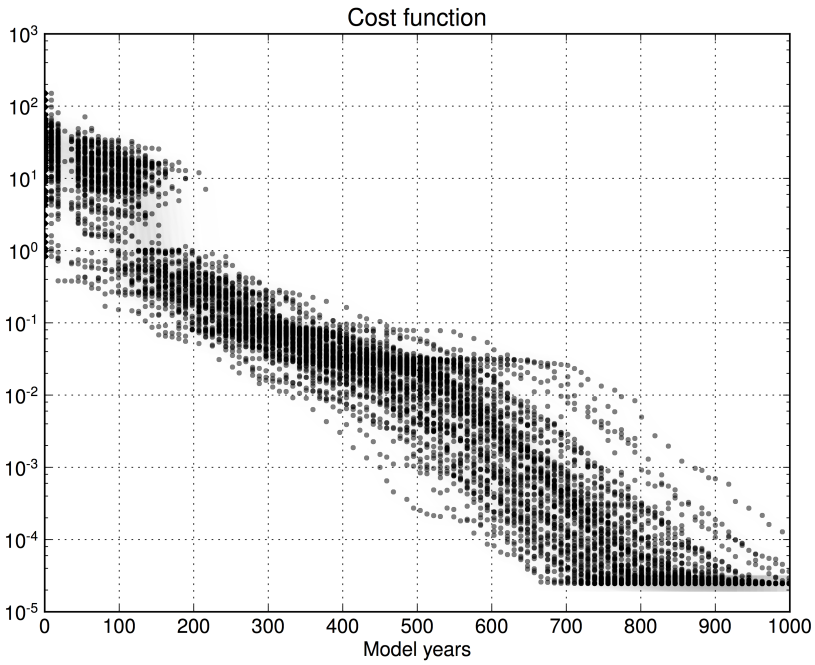


Figure 2.6: Decay of the cost function. Shown are the results of all twin experiments. Each depicted marker is an iterate of an optimization process.

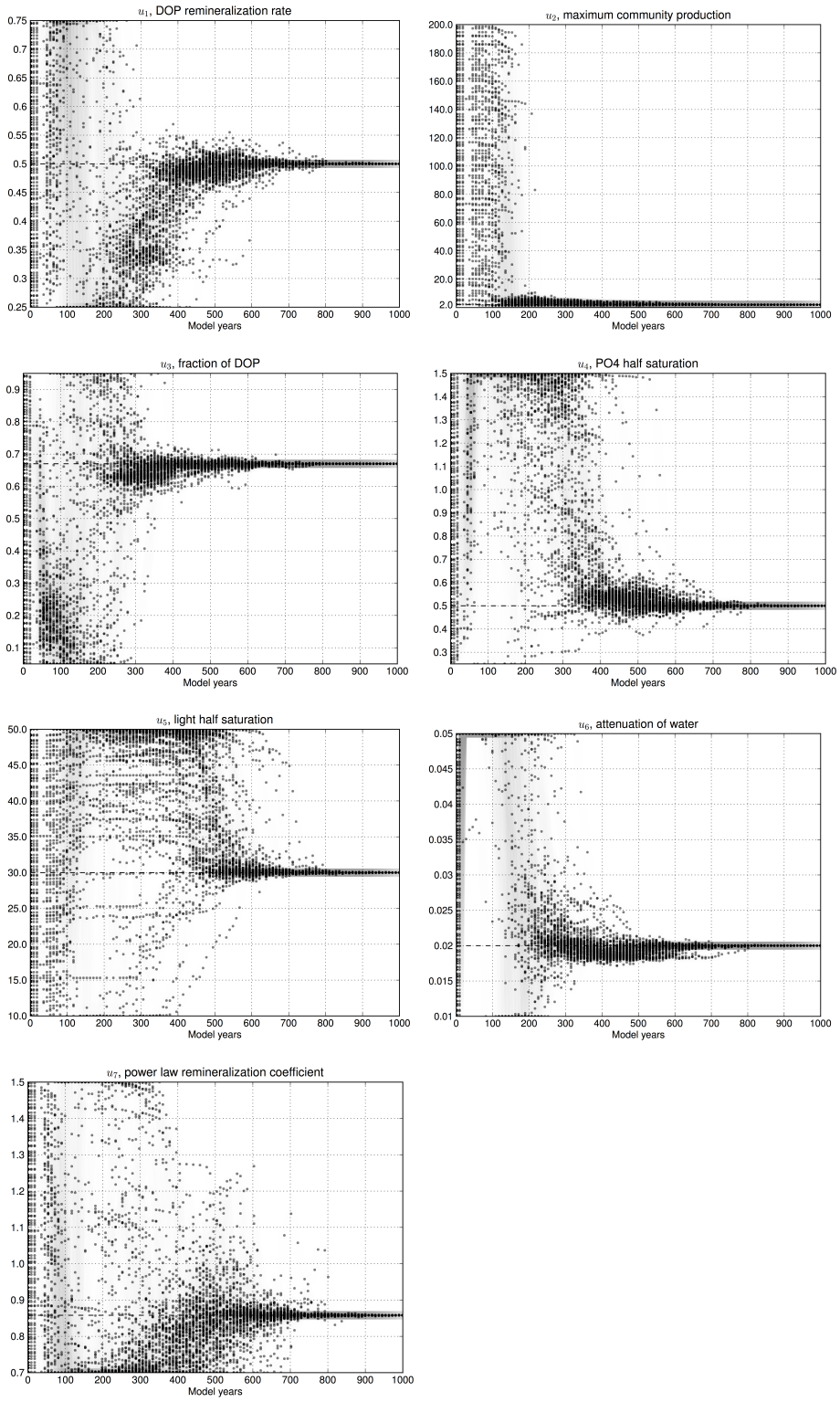


Figure 2.7: Convergence of the model parameters towards their reference values. Shown are the results of all twin experiments. Each depicted marker is an iterate of an optimization process.

APPENDIX

Journal of Computational Methods in Sciences and Engineering 12 (2012) 47–62
DOI 10.3233/JCM-2012-0403
IOS Press

Parameter identification in climate models using surrogate-based optimization

M. Prieß[a,*], J. Piwonski[a], S. Koziel[b] and T. Slawig[a]
[a]*Institute for Computer Science, Cluster The Future Ocean, Christian-Albrechts Universität zu Kiel, Kiel, Germany*
[b]*Engineering Optimization and Modeling Center, School of Science and Engineering, Reykjavik University, Reykjavik, Iceland*

Abstract. We present initial steps and first results of a surrogate-based optimization (SBO) approach for parameter optimization in climate models. In SBO, a computationally cheap, but yet reasonably accurate representation of the original high-fidelity (or fine) model, the so-called surrogate, replaces the fine model in the optimization process. We choose two representatives, namely two marine ecosystem models, to verify our approach. We present two ways to obtain a physics-based low-fidelity (or coarse) model, one based on a coarser time discretization, the other on an inaccurate fixed point iteration. Since in both cases, the low-fidelity model is less accurate, we use a multiplicative response correction technique, aligning the low- and the high-fidelity model output to obtain a reliable surrogate at the current iterate in the optimization process. We verify the approach by using model generated target data. We show that the proposed SBO method leads to a very satisfactory solution at the cost of a few evaluations of the high-fidelity model only.

Keywords: Climate models, marine ecosystem models, parameter identification, parameter optimization, surrogate-based optimization, low-fidelity models, response correction

1. Introduction

Modeling the Earth's climate system and performing numerical simulations to forecast its future behavior is one of the most challenging tasks in applied mathematics and other scientific disciplines. There are several reasons, among them the huge complexity of the system due to the many interactions between atmosphere, oceans, biosphere and cryosphere (sea and land ice), and finally the human impact.

The processes to be modeled and simulated are ranging from fluid mechanics (in atmosphere and oceans) to bio- and biochemical interactions, e.g., in marine or other type of ecosystems. Moreover, the spatial and temporal scales are quite different. There are non-negligible effects on very small scales (e.g., influence of clouds in the atmosphere and turbulence in the ocean), but on the other hand some processes take a very long time. This is the case, for example, for the large-scale currents in the ocean. Thus a numerical model has to be run (or "spin-up" as it is sometimes called in the climate community) to produce a steady or stable and meaningful output such that forecasts are possible.

Another important point is that there are no really stationary states in the system, there is at least an annual seasonal and moreover a daily cycle due to the impact of the sunlight as *the* major driving force

*Corresponding author: M. Prieß, Institute for Computer Science, Cluster The Future Ocean, Christian-Albrechts Universität zu Kiel, 24098 Kiel, Germany. Tel.: +49 431 880 7452; Fax: +49 431 880 7618; E-mail: mpr@informatik.uni-kiel.de.

of the system. Thus, the models are usually not stationary, but time-dependent. In several applications, steady periodic solutions representing the annual cycle are computed.

As a consequence, and since many important processes are non-linear, the numerical effort to simulate the whole or parts of the climate system with a satisfying accuracy and resolution is quite high. It usually involves high performance computing. This is the reason why many kinds of reduced order models or (again in the notion of climate research:) "parametrizations" are used to reduce the system size and thus the computational effort, see e.g. [13]. One typical example how this can be done is the modeling of the daily cycle by periodic functions. In these parametrizations several additional unknowns (the parameters) enter the system. Many of them are not known beforehand and not directly measurable. They have to be identified using measurement data. This is the point where large-scale optimization methods become crucial for a climate system forecast. Before a transient simulation of a model is possible, the latter has to be calibrated and validated, i.e., the relevant parameters have to be identified and ideally uncertainties have to be quantified.

The mathematical task of parameter identification using optimization methods can be classified as a least-squares type optimization or inverse problem [2,3,24]. The number of optimization parameters range from about 10 to 100 real-valued ones (e.g., in our marine ecosystem model examples below) up to distributed functions with thousands and more parameters after discretization (for example when an initial model state or boundary condition is unknown). Additionally, constraints on the parameters (e.g., non-negativity of growth-rates in ecosystem models etc.) and on the state variables (non-negativity of concentrations of biological species as algae etc. or of temperature) might be given.

Mathematical optimization based on simulation models, no matter whether deterministic/local or stochastic/global methods are used, usually leads to a computational effort which is by a factor higher than the one of the pure simulation. This is the reason why any kind of method that reduces this optimization effort is highly appreciated.

The method we use here for a typical class of climate models is the so-called *Surrogate-based Optimization* (*SBO*) approach. It is widely and very successfully used in engineering sciences, compare [1, 7,11,18]. In this approach, the idea is to introduce a surrogate, i.e., a computationally cheap and yet reasonably accurate representation of the original, in the SBO-terminology called *high-fidelity model*. The surrogate replaces the high-fidelity model in the optimization process in the sense of providing predictions of the optimal parameters. Additionally, it is updated using the high-fidelity model output accumulated during the optimization process. This scheme of predicting and updating is normally iterated in order to refine the search and to locate the high-fidelity model optimum as precisely as possible.

One important task in the SBO method is the choice of the surrogate itself, for which there are several opportunities: It can be either created by using sampled high-fidelity model output and working on this *response surface* of the original model. Another way that we follow here is to employ a physics-based, so-called *low-fidelity* or *coarse model*. The low-fidelity model is normally less accurate. Therefore, it has to be iteratively corrected by suitable methods. The correction or alignment can be realized using a limited number (in many cases, only one) evaluations of the high-fidelity model and possibly also its derivatives. Compared to a "direct" optimization of the high-fidelity model, the SBO approach can be very efficient in terms of number of function evaluations of the high-fidelity model necessary to yield a reasonably accurate (sub-)optimal solution.

The question remains how to construct the low-fidelity model in this case. This clearly depends on the underlying model equations. Climate models and also marine ecosystem models that we use here as a representative sub-class, are typically given as time-dependent partial differential or differential algebraic equations (PDE/PDEAs), compare [8,12,13]. One straightforward way to introduce a low-fidelity model

for the SBO method thus is to reduce the spatial or temporal resolution. The latter is used in a spatially one-dimensional model example below. Another opportunity to obtain the low-fidelity model that we show for a 3D application is a reduced terminal accuracy in an iterative scheme used to compute a steady periodic solution.

The development and use of low-fidelity models obtained by e.g. coarser discretizations or by parametrizations is common in climate research [13]. However, the use of even more accurate surrogate models to speed up the optimization process is new.

There are also processes in the climate system where even without much simplification several quantities or parameters are unknown or very difficult to measure. This is for example the case for growth and dying rates in marine ecosystem models [5,20], one of which is taken as a test case for the surrogate-based optimization approach we analyze in this paper. Marine ecosystem models describe photosynthesis and other biogeochemical processes in the marine ecosystem that are important, e.g., to compute and predict the oceanic uptake of carbon dioxide (CO_2) as part of the global carbon cycle [20]. We point out that the mathematical formulation of the climate models we use is quite general, such that our approach is not limited to them but remains applicable for a wide range of time-dependent models.

The structure of the paper is as follows: In Section 2 we first briefly describe the general formulation of climate models and marine ecosystem models, which we study as examples. In Section 3 we present the parameter optimization problem. Thereafter, we shortly introduce the basic idea of surrogate-based optimization. In the next sections, we describe two ways we follow in this paper to construct low-fidelity models and the alignment to create the surrogate. We end the paper with our numerical results and short conclusions.

2. Marine ecosystem models as representatives of climate models

In this section, we give the basic structure of marine ecosystem models which serve as typical examples of climate models that are subject to parameter identification. Since the SBO approach is not restricted to the ecosystem models, we start with a quite general formulation of climate models to emphasize the generality of the SBO approach. We then highlight the special properties of marine ecosystem models, give two examples of the models we actually use in the SBO, and briefly describe the numerical schemes used. They are important for our choice of the low-fidelity models to obtain the surrogate.

2.1. General formulation of climate models

As already mentioned above, climate models can be quite generally written as coupled systems of PD (A)Es, for example in the following form:

$$\left.\begin{aligned} E\,\frac{\partial y}{\partial t} &= f\left(y, \frac{\partial y}{\partial x_i}, \frac{\partial^2 y}{\partial x_i \partial x_j}, u\right) && \text{in } \Omega \times (0,T) \\ y &= y_{init} && \text{in } \Omega \\ By &= 0 && \text{on } \partial\Omega \times (0,T). \end{aligned}\right\} \tag{1}$$

Here $\Omega \subset \mathbb{R}^d, d \in \{1,2,3\}$, is the spatial domain with boundary $\partial\Omega$, T the end of the considered time interval, and y the vector of the model output or *state variables*. The function y_{init} describes initial data, and B is an optionally nonlinear boundary operator, for example the first normal derivative when representing a Neumann boundary condition. In this compact notation, the right-hand side f includes

all spatial differential operators as well as the coupling between the components of the state variable y. Typically, f is nonlinear and includes a vector of model parameters to be identified, here denoted by u.

The matrix E allows here for including algebraic equations in the system (then being singular). We include PDAEs in this formulation since in climate models, e.g., ocean circulation models (see [8]), the Navier-Stokes equations are an important part, and – when written in the above form – are a PDAE system. In our example of marine ecosystem models (which are formulated as PDE system), the matrix E can be set to the identity and thus omitted.

2.2. Marine ecosystem models

Marine ecosystem models mainly consist of two parts, namely the ocean circulation and the biogeochemical model. For the investigation of the oceanic CO_2 uptake especially the latter is subject of current research, see [14]. Ocean biota plays an import role within the global carbon cycle, but its complex organic and inorganic cycles are challenging when formulating a comprehensive biogeochemical model, see [20].

The coupling between ocean circulation and the biogeochemical interactions as photosynthesis, dying and growth of plankton species etc. is mostly regarded as one-way coupling. This means that the influence of the circulation (including temperature and maybe salinity distribution) on the biota is assumed to be much higher as vice versa and thus is considered only. This is often called an *off-line computation*: Velocity and temperature fields are computed beforehand by an ocean circulation model and only used as forcing data for the biogeochemical simulations. It becomes clearly attractive since the amount of computation is reduced significantly. Such an offline computation is especially appropriate when considering so-called *passive tracers*, i.e. biogeochemical species that have no influence on the ocean circulation or temperature distribution. By using pre-computed circulation data, all tracers necessarily are regarded as passive.

In our example models, we use this off-line mode. The model equations then form a system of coupled transport or advection-diffusion-reaction equations, where the reaction terms are given by the biogeochemical processes. We may write our model equations now as a special form of (1), where the first equation is given by

$$\frac{\partial y_i}{\partial t} = \operatorname{div}(\kappa \nabla y_i) - \operatorname{div}(v y_i) + q_i(y, u), \quad i = 1, \ldots, n.$$

Here $y = (y_i)_{i=1,\ldots,n}$ is the vector of tracers/ state variables, and $y_i = y_i(t, x)$ denotes a single tracer concentration at time t in $x \in \Omega$. We assume homogeneous Neumann boundary conditions on $\partial\Omega$ for all tracers. The time dependent turbulent mixing/diffusion coefficient κ and the velocity vector field v are precomputed data from an ocean model and thus *not* subject to identification here. Note that the turbulent mixing dominates the molecular tracer diffusion in this application, and thus κ is the same for all tracers.

The parameters to be identified are coefficients in the nonlinear biogeochemical coupling terms $q_i(y, u)$. They are similar to non-autonomous predator-prey models, since for example the growth rate of phytoplankton/algae depends on the sunlight and thus on space and time. Nearly all of these terms are local, i.e. they describe processes happening at point x and not depending on neighborhood points. Some of them include sinking processes and thus become non-local. The sinking velocity of dead material for example is one parameter that has often to be identified. Note that in some of the q_i also the precomputed temperature of the water enters, which we omitted here for brevity. It is also not target of the identification process in this paper.

2.3. Two examples for biogeochemical models

In this section we briefly give two examples for biogeochemical models, namely the ones we studied using the SBO approach. There is not *the* biogeochemical model, thus modelers are interested not only in the right parameters for a single model, they moreover try to figure out what is the "right" one and especially how to choose an appropriate or optimal number and subset of tracers and species representations.

2.3.1. N-DOP model

The N-DOP model admits phosphate (nutrients, N) and dissolved organic phosphorus (DOP). The tracers are denoted by $y = (y_1, y_2) = (y_N, y_{DOP})$. A main and typical feature of the model is that the biogeochemical terms q_i differ in two horizontal layers or zones, namely the upper, *euphotic zone* Ω_1 (where light enables the photosynthesis) and the lower, non-euphotic zone Ω_2 of the ocean. Thus we obtain the following coupling terms:

$$q_1(y,u) = \begin{cases} -g(y_1, I) + \lambda y_2 & \text{in } \Omega_1 \\ \bar{\sigma}\partial_z G(y_1, I) + \lambda y_2 & \text{in } \Omega_2 \end{cases}$$

$$q_2(y,u) = \begin{cases} \sigma\, g(y_1, I) - \lambda y_2 & \text{in } \Omega_1 \\ -\lambda y_2 & \text{in } \Omega_2. \end{cases}$$

The vertical spatial coordinate is denoted by z in this model, and ∂_z is the corresponding partial derivative. We have summarized here – as above – all parameters that appear (see below) in the vector u on the left-hand side. The nonlinear function

$$g(y_1, I) = \alpha \frac{y_1}{y_1 + K_1} \frac{I}{I + K_I} \tag{2}$$

is the *biological production* which is calculated depending on nutrients and light I. It is limited using a so-called *half saturation function* and a maximum production rate parameter α. Light is computed from the short wave radiation (as a function of latitude and time), the photosynthetically available radiation, the ice cover and the exponential attenuation of water K_{H2O}, which is – as $\lambda, \sigma, \bar{\sigma}, \alpha, K_1$, and K_I – another parameter to be identified. A fraction σ of the biological production remains suspended in the water column as dissolved organic phosphorus, which remineralizes with a rate λ. The remainder of the production sinks as particulate to the bottom where it is remineralized according to an empirical power law relationship

$$G(y_1, I) = (z/l)^{-b} \int_0^l g(y_1, I)\, d\xi.$$

Here, b is another parameter, such that we have $u = (\lambda, \sigma, \bar{\sigma}, \alpha, K_1, K_I, b)$. For further model details we refer to [15].

2.3.2. NPZD model

The NPZD model by Oschlies and Garcon [14] is a system for four tracers $y = (y_1, \ldots, y_4) = (N, P, Z, D)$ with N (nitrogen), P (phytoplankton), Z (zooplankton), and D (detritus). The coupling

terms are given by

$$
\begin{aligned}
q_1(y,u) &= \gamma_2 y_3 + \mu_D y_4 - J(y_1,y_2,z,t)y_2,\\
q_2(y,u) &= J(y_1,y_2,z,t)y_2 - G(y_2)y_3 - \mu_P y_2,\\
q_3(y,u) &= \gamma_1 G(y_2)y_3 - \gamma_2 y_3 - \mu_Z y_3^2\\
q_4(y,u) &= (1-\gamma_1)G(y_2)y_3 + \mu_P y_2 + \mu_Z y_3^2 - \mu_D y_4 - w_s \partial_z y_4.
\end{aligned}
$$

Again, z is the vertical coordinate. The system involves also quadratic terms, an explicit sinking velocity w_s for detritus and a non-differentiability, namely in the growth rate of phytoplankton (y_2 or P) which is modeled as

$$J(y_1,y_2,z,t) = J_{\max}(z,t)\min\{J_1(y_1), J_2(y_2,z,t)\}.$$

It is modeled similar to (2) including two terms J_1, J_2 which describe the dependency on nutrients and light, respectively. Their form is similar to both terms in (2). The difference in this model is that not the product of both terms is used, but the minimum, compare [14,20] for further discussion and motivation. This modification makes the model non-differentiable. As a second difference to the N-DOP model above the maximum growth rate

$$J_{\max}(z,t) = \mu_m C_{ref}^{c\Theta(z,t)}, \quad \mu_m, c, C_{ref} \in \mathbb{R},$$

depends on the water temperature Θ, which has to be provided by an ocean circulation model. Another nonlinear term in the equations is the zooplankton grazing function

$$G(y_2) = \frac{g\epsilon y_2^2}{g + \epsilon y_2^2}$$

which describes the transfer from phytoplankton to zooplankton and detritus. There are totally twelve model parameters to be identified here, among them are $\gamma_1, \gamma_2, \mu_D, \mu_P, \mu_Z, \mu_m, w_s, g, \epsilon$.

2.4. Numerical solution

Since almost all climate models given in the form (1), their are solved via a time-stepping scheme. In the ecosystem models, usually the diffusion part is solved implicitly, and the biogeochemical reaction and the advection part explicitly. In the off-line computations we study here, there are two ways to make use of the precomputed ocean circulation data (velocity vector field v and scalar, but also spatially and temporally varying diffusion κ):

- The data can be stored directly and afterwards used for assembling the system matrices for the differential operators for diffusion and advection in the biogeochemical model itself (this is done in the NPZD model).
- The ocean model that precomputes the data can be used to generate the necessary, so-called *transport matrices*, see [9]. These matrices usually represent a mean ocean circulation field for one month. In this approach, for the discretization of the tracer transport the numerical scheme of the underlying ocean model is used, which makes this approach very flexible (with respect to the choice of the biogeochemical coupling terms q_i) and attractive when testing the behavior of different biogeochemical models. This approach is used for the N-DOP model. It is realized in the software package

METOS3D, see [16]. In our example, the underlying ocean model uses an operator splitting technique that discretizes part of the spatial derivatives implicitly, part of them explicitly. The main feature of the transport matrix approach is that for time integration of the tracer transport the numerical scheme of the ocean model that was used to generate the matrices can be treated as a black box. The biogeochemical model part is always treated by an explicit Euler method with constant step-size. The total number of degrees of freedom in space was about 60'000, the time-step can vary between 3 and 192 hours model time.

Another difference between our two examples for the SBO approach lies in the setup of the numerical simulation that is target of the optimization: One may consider a complete transient run with time-dependent forcing data (as for example velocity or temperature) or a spin-up into a steady quasi-periodic or periodic seasonal cycle, thus applying some kind of fixed point iteration. For the NPZD model we study the transient simulation, and for the N-DOP model the steady periodic case, which involves about 3000 years of model time to converge. This choice is independent of the selection of the two models, it could as well be vice versa. The fixed point iteration is also performed using METOS3D.

And, finally, a third difference is the spatial dimension: Whereas for the NPZD model we only compute in a single water column at a fixed location, we perform a complete 3D run for the N-DOP model, taking into account the ecosystem in the whole ocean.

3. A least squares optimization for the solution of the inverse problem

In this section we formulate the optimization problem that was used to identify the model parameters, i.e., to solve the inverse problem. Note that we consider the discretized model output and also real-valued parameters only. The model output, i.e., the values of all tracers at the points in the space-time-grid, we denote by bold-faced $\mathbf{y}$. That means that $\mathbf{y}$ represents a n-dimensional object (n again being the number of species or tracers in the ecosystem), where every component is the trajectory of a one- (for the NPZD model) or three-dimensional (for the N-DOP model) tracer concentration.

Given some observational or synthetic target data $\mathbf{y}_d$ and reasonable bounds b_l, b_u on the model parameter vector u, the optimization problem now can be written as

$$\min_{u \in U_{ad}} J(\mathbf{y}(u)) \tag{3}$$

where $\mathbf{y}(u)$ is a solution of the corresponding discretized state equation with given parameter vector u and

$$J(\mathbf{y}) := \frac{1}{2}||\mathbf{y} - \mathbf{y}_d||^2, U_{ad} := \{u \in \mathbb{R}^m : b_l \leqslant u \leqslant b_u\}$$

Here we denote the total number of model parameters to be identified by m. The inequalities in the definition of the set U_{ad} of admissible parameters are meant component-wise. The functional J may additionally include a regularization term for the parameters, which was not necessary in our case. The norm in the cost function is a weighted Euclidean norm where the weights may be taken as the inverse values of the variances of the measurements, see [19] for an example for the NPZD model. We performed our SBO experiments here with synthetic data generated by the models themselves, thus these weights are not that important.

4. Surrogate-based optimization

In this section we describe the basic idea of the SBO approach. Surrogate-based optimization (SBO), see [1,7,11,18], replaces the original or direct optimization cycle

simulation (high fidelity) $\rightarrow$ parameter update $\rightarrow$ simulation (high fidelity) $\rightarrow \cdots$

where all simulations are done with the original, computationally expensive *high-fidelity model*, by another cycle that can be sketched as

simulation (high-fidelity) $\rightarrow$ construction/update of surrogate
$\rightarrow$ optimization of surrogate
$\rightarrow$ parameter update $\rightarrow$ simulation (high-fidelity) $\rightarrow \cdots$

In both cases, one simulation may also include sensitivity computations. Using SBO, the computational effort can be significantly reduced, since every iteration involves one evaluation of the high-fidelity model only (plus optionally its sensitivity). Furthermore, optimizing the surrogate requires just evaluations of the computationally cheaper low-fidelity model.

A natural consequence of this idea is that the surrogate should be computationally cheap. The approach that we use in this paper is to construct the surrogate through correction/alignment of a physics-based low-fidelity or coarse model, a less accurate but computationally cheap representation of the original one. These so-called *physics-based surrogates* [23] typically inherit main physical characteristics of the original model. Another choice to build the surrogate – that we do not discuss here – would be to use *functional surrogates*, where approximations of the high-fidelity model output is used (e.g., by polynomial regression [18], kriging [21] or support-vector regression [22]). One possible way to create a physical low-fidelity or coarse model is by using a coarser discretization in space and/ or time. We use this method for the NPZD model by coarsening the time discretization while retaining stability of the numerical scheme. This can usually be implemented by just a few changes or even just by changing one numerical parameter in the simulation code. In the case where an iterative scheme is used to compute a steady (stationary or periodic) solution to the model equations, there is additionally the option to stop the iteration after fewer steps and thus with less accuracy than the original one. We use this approach for the N-DOP model. It is obvious that the realization of this low-fidelity model is rather simple. Another approach that we do not use here is to evaluate a different, but simpler model that maybe omits some processes or is linearized.

The surrogate is constructed using the coarse model by reducing misalignment between the coarse and the fine model output. The specific correction technique exploited in this work is described below. Then the surrogate is optimized, i.e. in the kth step of the overall optimization a solution

$$u_{k+1} = \operatorname*{argmin}_{u \in U_{ad}} J\left(\mathbf{s}_k(u)\right) \tag{4}$$

is computed. This optimization is much cheaper than the original one, since in every step of this inner iteration, only a coarse model evaluation and a constant correction (which is also cheap) is performed. When the solution is obtained, a fine model output (and optionally its sensitivity) is computed for the new parameter vector u_{k+1}, the surrogate is updated and again optimized. This process is iterated.

If the surrogate $\mathbf{s}_k$ satisfies so-called 0-order and 1st-order consistency conditions [4,10] with the fine model at u_k, i.e.,

$$\mathbf{s}_k(u_k) = \mathbf{b}(u_k), \quad \mathbf{s}_k'(u_k) = \mathbf{b}'(u_k), \tag{5}$$

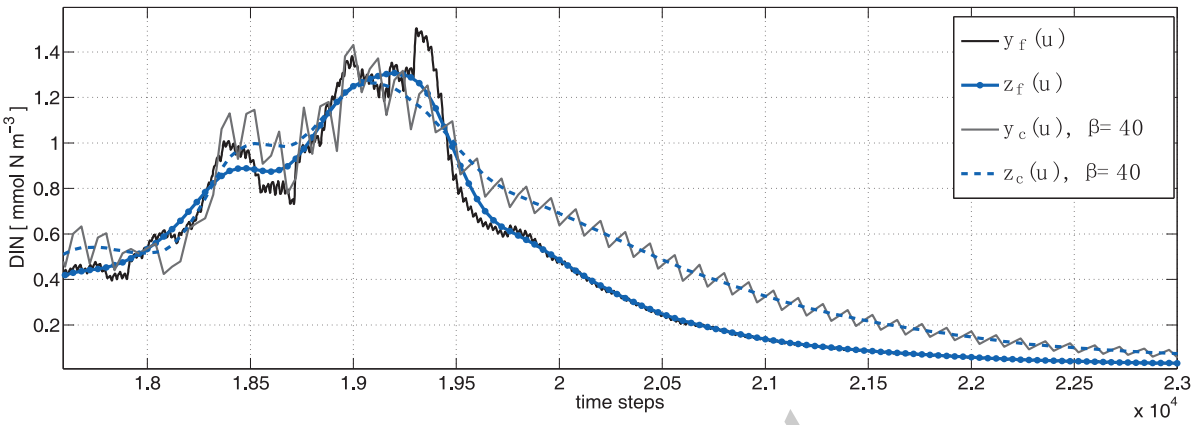


Fig. 1. Comparison of high- and low-fidelity output, both "raw" ($\mathbf{b}_f$, $\mathbf{b}_c$) and smoothed ($\mathbf{z}_f$, $\mathbf{z}_c$), for the tracer dissolved inorganic nitrogen (N) for one year of the NPZD model output. Shown is the uppermost layer in the water column.

the whole SBO scheme is provable convergent to at least a local optimum of Eq. (3), provided that both the coarse and the fine model is sufficiently smooth, and the surrogate optimization step is enhanced by the trust-region safeguard [4,10], i.e., in (4) a local solution in the ball around u_k with a given trust-region radius is computed. Note that 1st-order consistency requires evaluation of fine model sensitivity at each iteration of the SBO. Clearly, for a given problem, the trade-offs between the solution accuracy and the extra computational overhead related to sensitivity calculation would have to be investigated.

In this work, the surrogate is defined to satisfy the 0-order consistency only which is sufficient to ensure good optimization performance for synthetic data.

5. Low-fidelity models

In this section we describe the construction of the low-fidelity (or coarse) models that we use for the two example applications. From now on, the coarse model output will be denoted by $\mathbf{y}_c$ to be distinguishable from the fine model one, denoted by $\mathbf{y}_f$.

5.1. Coarsening the time discretization

It is quite obvious that a coarser time discretization allows us to obtain a computationally cheaper, but less accurate model. Choosing a coarser time discretization is appropriate whenever a transient solution is regarded as model output that is compared to the data or desired state $\mathbf{y}_d$ in the optimization, and no spin-up into a steady state is performed. For the NPZD model this approach was used. For details on the numerical scheme that is used for the high-fidelity model, we refer to [17]. Denoting by τ the original time step of the model, the numerical solution of the low-fidelity model is obtained by employing

$$\hat{\tau} = \beta\tau,$$

accordingly, with a coarsening factor $\beta \in \mathbb{N} \setminus \{0, 1\}$, while keeping the spatial discretization fixed.

It is important to keep in mind that choosing β too large could lead to a numerically unstable scheme [6]. The condition on stability is determined by the ratio h/u_1 and the nonlinear coupling term q, where h denotes the spatial step-size. More specifically, we choose $\beta = 40$, which in numerical tests as well as

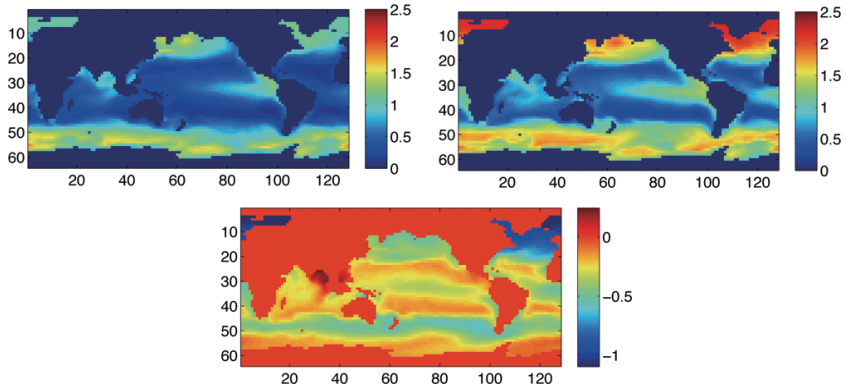


Fig. 2. High-fidelity (after 3000 fixed point iterations, top left), low-fidelity model output (after 25 iterations, top right) and difference for the tracer N, for the top ocean layer and at the first time step in the year.

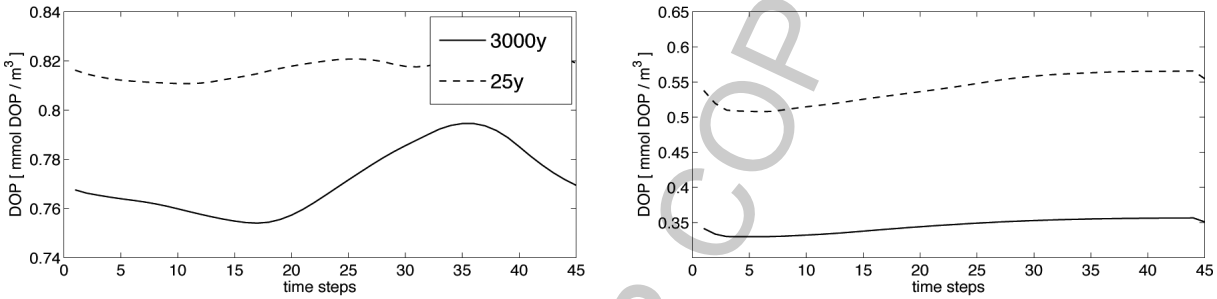


Fig. 3. Trajectories of the last year/iteration for the high-fidelity (left) and the low-fidelity model output (after 25 iterations) for the tracer DOP, at two illustrative spatial locations.

in a theoretical study that we omit here proved to retain stability. The differences in the model output is depicted in Fig. 1.

Results presented in [17] validate for the NPZD model that such a low-fidelity model if further corrected by a suitable response correction technique, leads to a reliable approximation of the high-fidelity model.

5.2. *Relaxed fixed point convergence*

When the model output is a steady stationary or periodic state and a fixed point iteration is applied, a straight-forward way is to reduce the number of iterations therein to obtain a low-fidelity model. The implementation of this approach is very simple. It just requires changing the stopping criterion. Furthermore, since in this case, both the fine and the coarse model grids are identical, many issues in the formulation of the surrogate and subsequent optimization are rather simplified, such as the formulation of the response correction and of the cost functions as well as comparability of the solution of the SBO. We used this method for the three-dimensional N-DOP model with a reduced number of 25 fixed point iterations instead of the original 3000 that are applied in the original high-fidelity model. The numerical effort for one model evaluation is thus reduced by a factor of 120. One fixed point iteration is equivalent to one year model time and itself consists of 45 time steps.

As motivation for this choice, Figs 2 and 3 show the difference between the solutions (horizontal tracer distribution in the uppermost layer in Fig. 2 and trajectory of the last year at two different spatial

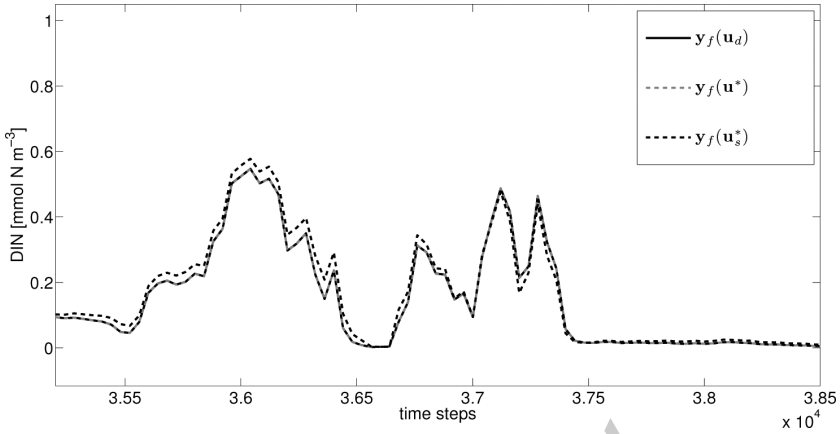


Fig. 4. Comparison of the desired state and the fine model output, for the NPZD model, for the solutions of a direct fine optimization u^* and the SBO solution u_s^*. Shown is again the tracer dissolved inorganic nitrogen (N), for one year of the model output and for the uppermost layer in the water column.

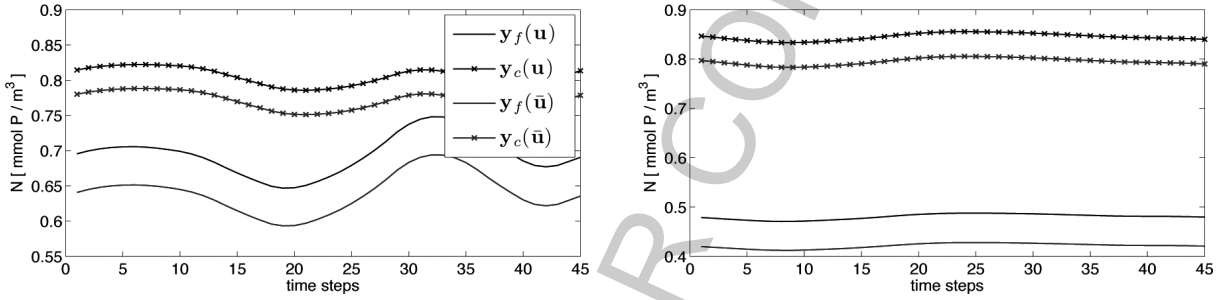


Fig. 5. Whole trajectories (one year) of the fine and the coarse model output at a reference parameter $u = u_k$ and a neighboring point $\bar{u}$, here, for the tracer N in the first layer at two spatial locations (left: index pairs in horizontal directions (32, 32), right: (117, 43)).

coordinates in Fig. 3). It can be seen that there is a significant difference, but an overall similarity in structure of the solutions of the high-fidelity (converged) solution and the low-fidelity (not fully converged) one where the fixed point iteration was stopped after 25 steps.

6. Surrogate construction

In this section we describe how the surrogate $\mathbf{s}_k$ at iteration k of the overall parameter identification process is generated. The surrogate uses the low-fidelity/coarse and the high-fidelity/fine model outputs $\mathbf{y}_c$ and $\mathbf{y}_f$, respectively, and computes a point-wise (in space and time) alignment by the following *multiplicative correction*

$$s_{k,j}(u) := a_{k,j}\; y_{c,j}(u), a_{k,j} \;:=\; \frac{y_{f,j}(u_k)}{y_{c,j}(u_k)},$$

where k is the outer iteration of the SBO process and j the index of the discretized state variable vector $\mathbf{y}$, depending on the tracer index, the time step and the point in the spatial grid. We call $a_k := (a_{k,j})_j$

the *correction vector* in step k. This correction scheme is justified by the fact that the overall "shape" of the coarse model response resembles that of the fine one. Furthermore, the qualitative relation between the fine and the coarse model output is well preserved even when moving from one parameter vector u_k to another (see for example Fig. 5).

By definition, this surrogate ensures 0-order consistency with the fine model (i.e., agreement in the function values). Further enhancements of this formulation by using fine/coarse model sensitivity data are possible, to satisfy also the 1st-order consistency (i.e., agreement in the function values). This latter was not used in this paper. However, a zero-order consistent surrogate might still be able to yield a sufficiently accurate solution, since the physics-based surrogate inherits substantial knowledge about the marine model under consideration, its derivatives are expected to be at least similar to those of the fine model. Note that the surrogate model as defined above is constructed using just one evaluation of the fine model which keeps the optimization costs low. Clearly, including fine model sensitivity increases this cost. Thus, for a given problem, the trade-offs between the solution accuracy and the extra computational overhead related to sensitivity calculation would have to be assessed.

7. Numerical results and discussion

In this section we present the results obtained by exemplary surrogate-based optimization runs for the two marine ecosystem models, employing the proposed response correction technique to the selected coarse models. For all optimization runs, we used the MATLAB[1] function `fmincon`, exploiting the active-set algorithm. For the NPZD model, where a transient run is performed in one space dimension, the results have been shown in detail in [17]. Below, we provide a brief summary. For the N-DOP model in 3D, where a steady periodic solution is computed, we present initial numerical results, which indicate the applicability of the proposed method.

7.1. NPZD model using coarser time discretization

As has been demonstrated in [17], the relationship between the fine and coarse model response indicates that the natural way of constructing the surrogate would be the proposed multiplicative response correction. It was furthermore shown that "smoothing" of the coarse model response is reasonable (due to a rather inaccurate model response when employing a large time step in its numerical solution) such that the resulting smoothed response contains the relevant features of the fine one (cf. Fig. 1). For the proposed response correction, in order to obtain a commensurable fine model response, the latter is smoothed accordingly.

It has been demonstrated that the proposed SBO technique is able to obtain a sufficiently accurate solution at the cost of a few fine model evaluations only. A significant reduction in the optimization costs of about 84% could be obtained, when compared to a direct fine model optimization.

Note that for the optimization runs, ideal, model generated data has been utilized to create a synthetic target, whereas, for the initial investigations, we did not consider noisy data. For a discussion of the latter, see for example [19].

Figure 4 shows an illustrative trajectory of the desired state $\mathbf{y}_d$, the fine model $\mathbf{b}_f$ output at the solution u^* of a direct fine model optimization and at the solution u_s^* obtained by SBO. Table 1 presents the

[1]MATLAB is a registered trademark of The MathWorks, Inc., http://www.mathworks.com.

Table 1
Target and optimal parameters $\mathbf{u}_d, \mathbf{u}^*, \mathbf{u}_s^*$, for an illustrative fine and SBO run

Iterate	u_1	u_2	...									u_{12}
$\mathbf{u}^*$	0.747	0.596	0.025	0.01	0.03	0.999	2.046	0.01	0.203	0.02	0.493	4.31
$\mathbf{u}_s^*$	0.705	0.626	0.044	0.015	0.06	0.937	1.908	0.016	0.147	0.02	0.629	4.237
$\mathbf{u}_d$	0.75	0.6	0.025	0.01	0.03	1.0	2.0	0.01	0.205	0.02	0.5	4.32

corresponding parameter values. It can be observed that fine model optimization is able to reconstruct the target almost perfectly, both in terms of matching the target response and optimal parameters, whereas the solution obtained by SBO is very accurate, most importantly, in terms of matching the target response. Clearly, the accuracy for some parameters obtained by SBO is not perfect for the shown optimization run (cf. Table 1), which is mostly because of low sensitivity of the model with respect to some of the parameters. This has already been discovered in [19]. However, improved matching between the optimized model parameters and those corresponding to the target output could be obtained by executing a larger number of SBO iterations.

We point out, that in those SBO runs we did not use fine model sensitivity data in the definition of the surrogate. Thus, first-order consistency (i.e., agreement in the first-order derivatives between the surrogate and the fine model response) is not ensured exactly. Formally, this is not sufficient to ensure the convergence of the surrogate-based scheme to a (local) minimum of the fine model optimization problem. However, as pointed out before, since the physics-based surrogate inherits substantial knowledge about the marine model under consideration, its derivatives are expected to be at least similar to those of the fine model. Furthermore, because of being constructed from a physics-based coarse model, the surrogate exhibits quite good generalization capability, which means thats it provides a reliable approximation of the fine model when moving from one parameter vector to another. Still, the use of derivative information together with trust-region convergence safeguards [4,10] would bring further improvement in terms of matching accuracy.

7.2. N-DOP model with inaccurate fixed point solver

For the N-DOP model, we consider a reference fine model where we employ 3000 fixed point iteration steps. A properly selected coarse model (here, using 25 fixed point iteration steps) and the proposed multiplicative response correction allows us to build a reliable surrogate. As initial approach, we did not employ coarse/fine model sensitivity data in the surrogate's construction. However, as for the NPZD model, the underlying coarse model is as well physics-based, such that the constructed surrogate again exhibits quite good generalization capability. For the considered coarse model, smoothing is not necessary.

The evaluation time for the selected coarse models is, compared to the one for the fine model, reduced by the factor 120. Whereas the fine model evaluation takes several minutes on a 48-processor cluster, the time required for one coarse model evaluation is significantly reduced to a few seconds. Since no fine/coarse model sensitivity is used, the cost for the construction of the surrogate is only one fine model evaluation here.

Straightforward attempts by employing the fine model of this computationally very expensive three-dimensional model under consideration directly in an optimization loop using conventional optimization algorithms are tedious, since typically a large number of expensive fine model evaluations is required. A well performing SBO would thus be able to dramatically reduce the overall optimization costs.

In order initially assess the performance of the proposed surrogate in a SBO scheme, we performed first optimization steps. Figures 6 and 7 shows the result after two iterations. It can be observed that SBO

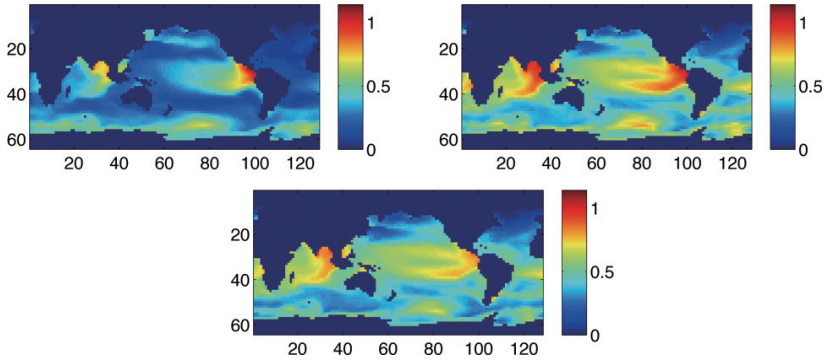


Fig. 6. Solution (here, N) of the N-DOP model (top layer) for an initial guess of the parameters (top left), the desired state with chosen parameter vector (top right), and the fine model output after two iterations in a SBO run.

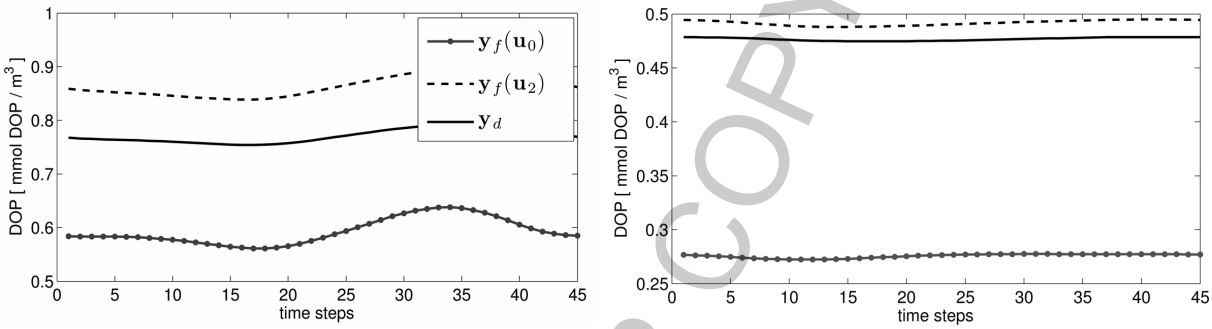


Fig. 7. Same as in Fig. 6, but trajectories for tracer DOP at two illustrative grid points.

is able to obtain a solution already significantly closer to the target which indicates that the surrogate provides a sufficiently accurate prediction of the fine model optimum. Therefore, its application in a full optimization run seems very promising.

Clearly, performing more comprehensive numerical tests and optimization runs will be necessary to show the applicability of the proposed SBO approach for this model. Furthermore, improvements of the present approach by including fine/coarse model sensitivity as well as the trade-offs between the solution accuracy and the extra computational overhead related to sensitivity calculation will be investigated.

8. Conclusions

Parameter identification in climate models can be very expensive in terms of the cost function and gradient evaluations, especially for three-dimensional cases. Accelerating this optimization process is therefore highly desirable. One possible way to comply with this aim is to replace the high-fidelity (or fine) model in the optimization run by a computationally cheap, but still reasonably accurate representation (surrogate).

The surrogate can be based upon a physical low-fidelity (or coarse) model which is a usually less accurate approximation and is hence furthermore aligned by suitable techniques. In this paper we consider two ways to construct this model for two different, typical applications. In both, the physical coarse model is obtained by employing the same simulation tool as for the fine model, whereas pursuing

a lower accuracy in the solution of the model state. For the alignment of this model to create a surrogate, we use a multiplicative response correction.

We successfully applied this methodology to the optimization of a one-dimensional marine ecosystem model and presented initial numerical results using a three-dimensional one, which indicated the applicability of the proposed method also for the computationally more expensive model.

We demonstrated that the surrogate-based optimization in conjunction with the chosen response correction approach and the coarse models under consideration is able to yield remarkably good results within a very few number of fine model evaluations only. Whereas a direct optimization approach using the fine model in a classical optimization loop – especially for three-dimensional applications – may require huge computational effort, the computational cost is significantly reduced.

Acknowledgments

The authors would like to thank Andreas Oschlies, IFM GEOMAR, Kiel. This research was supported by the DFG Cluster of Excellence Future Ocean.

References

[1] J.W. Bandler, Q.S. Cheng, S.A. Dakroury, A.S. Mohamed, M.H. Bakr, K. Madsen and J. Søndergaard, Space mapping: The state of the art, *IEEE T Microw Theory* **52**(1) (2004).
[2] H.T. Banks and K. Kunisch, *Estimation Techniques for Distributed Parameter Systems*, Birkhäuser, 1989.
[3] H. Martin Bücker, O. Fortmeier and M. Petera, Solving a parameter estimation problem in a three-dimensional conical tube on a parallel and distributed software infrastructure, *Journal of Computational Science* **2**(2) (5 2011), 95–104.
[4] A.R. Conn, N.I.M. Gould and P.L. Toint, *Trust-region methods*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2000.
[5] W. Fennel and T. Neumann, *Introduction to the Modelling of Marine Ecosystems*, Elsevier, 2004.
[6] C.A.J. Fletcher, *Computational Techniques for Fluid Dynamics*, volume 1, Springer, 2nd edition, 1991.
[7] A.I.J. Forrester and A.J. Keane, Recent advances in surrogate-based optimization, *Prog Aerosp Sci* **45**(1–3) (2009), 50–79.
[8] A.E. Gill, *Atmosphere – Ocean Dynamics*, volume 30 of *International Geophysics Series*, Academic Press, 1982.
[9] S. Khatiwala, M. Visbeck and M.A. Cane, Accelerated simulation of passive tracers in ocean circulation models, *Ocean Modelling* **9**(1) (2005), 51–69.
[10] S. Koziel, J.W. Bandler and Q.S. Cheng, Robust trust-region space-mapping algorithms for microwave design optimization, *IEEE T Microw Theory* **58**(8) (August 2010), 2166–2174.
[11] L. Leifsson and S. Koziel, Multi-fidelity design optimization of transonic airfoils using physics-based surrogate modeling and shape-preserving response prediction, *Journal of Computational Science* **1**(2) (6 2010), 98–106.
[12] A. Majda, *Introduction to PDE's and Waves for the Atmosphere and Ocean*, AMS, 2003.
[13] K. McGuffie and A. Henderson-Sellers, *A Climate Modelling Primer*, Wiley, 3nd edition, 2005.
[14] A. Oschlies and V. Garcon, An eddy-permitting coupled physical-biological model of the north atlantic. 1. sensitivity to advection numerics and mixed layer physics, *Global Biogeochem Cy* **13** (1999), 135–160.
[15] P. Parekh, M.J. Follows and E.A. Boyle, Decoupling iron and phosphate in the global ocean, *Global Biogeochemical Cycles* **19** (2005).
[16] J. Piwonski and T. Slawig, METOS3D*: A Marine Ecosystem Toolkit for Optimization and Simulation*, CAU Kiel, Institut für Informatik, http://www.informatik.uni-kiel.de/co2/software/meteos3d, 2011.
[17] M. Prieß, S. Koziel and T. Slawig, Surrogate-based optimization of climate model parameters using response correction, *Journal of Computational Science*, (0), 1877–7503, 2011.
[18] N.V. Queipo, R.T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan and P.K. Tucker, Surrogate-based analysis and optimization, *Prog Aerosp Sci* **41**(1) (2005), 1–28.
[19] J. Rückelt, V. Sauerland, T. Slawig, A. Srivastav, B. Ward and C. Patvardhan, Parameter optimization and uncertainty analysis in a model of oceanic CO_2-uptake using a hybrid algorithm and algorithmic differentiation, *Nonlinear Analysis B Real World Applications* **10**(1016) (2010), 3993–4009.
[20] J.L. Sarmiento and N. Gruber, *Ocean Biogeochemical Dynamics*, Princeton University Press, 2006.

[21] T.W. Simpson, J.D. Poplinski, P.N. Koch and J.K. Allen, Metamodels for computer-based engineering design: Survey and recommendations, *Eng Comput* **17** (2001), 129–150. 10.1007/PL00007198.
[22] A.J. Smola and B. Schölkopf, A tutorial on support vector regression, *Stat Comput* **14** (2004), 199–222. 10.1023/B:STCO.0000035301.49549.88.
[23] J. Søndergaard, *Optimization using surrogate models – by the space mapping technique*, PhD thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2003, Supervisor: Kaj Madsen.
[24] A. Tarantola, *Inverse Problem Theory and Methods for Model Parameter Estimation*, SIAM, 2005.

Geosci. Model Dev., 6, 17–28, 2013
www.geosci-model-dev.net/6/17/2013/
doi:10.5194/gmd-6-17-2013

Geoscientific Model Development

Porting marine ecosystem model spin-up using transport matrices to GPUs

E. Siewertsen[1], **J. Piwonski**[2], **and T. Slawig**[2]

[1]Institute for Computer Science, Christian-Albrechts Universität zu Kiel, 24098 Kiel, Germany
[2]Institute for Computer Science and Kiel Marine Science – Centre for Interdisciplinary Marine Science, Cluster The Future Ocean, Christian-Albrechts Universität zu Kiel, 24098 Kiel, Germany

Correspondence to: T. Slawig (ts@informatik.uni-kiel.de)

Received: 19 July 2012 – Published in Geosci. Model Dev. Discuss.: 31 July 2012
Revised: 7 November 2012 – Accepted: 5 December 2012 – Published: 8 January 2013

Abstract. We have ported an implementation of the spin-up for marine ecosystem models based on transport matrices to graphics processing units (GPUs). The original implementation was designed for distributed-memory architectures and uses the Portable, Extensible Toolkit for Scientific Computation (PETSc) library that is based on the Message Passing Interface (MPI) standard. The spin-up computes a steady seasonal cycle of ecosystem tracers with climatological ocean circulation data as forcing. Since the transport is linear with respect to the tracers, the resulting operator is represented by matrices. Each iteration of the spin-up involves two matrix-vector multiplications and the evaluation of the used biogeochemical model. The original code was written in C and Fortran. On the GPU, we use the Compute Unified Device Architecture (CUDA) standard, a customized version of PETSc and a commercial CUDA Fortran compiler. We describe the extensions to PETSc and the modifications of the original C and Fortran codes that had to be done. Here we make use of freely available libraries for the GPU. We analyze the computational effort of the main parts of the spin-up for two exemplar ecosystem models and compare the overall computational time to those necessary on different CPUs. The results show that a consumer GPU can compete with a significant number of cluster CPUs without further code optimization.

1 Introduction

This work is motivated by the usually huge effort that is needed when computing steady annual cycles (or, mathematically speaking, periodic solutions) of spatially three-dimensional marine ecosystem models. In most cases this is done by "spinning up" the model, i.e. by using a time-stepping algorithm with climatological, periodic forcing data until the steady cycle is reached, at least up to a certain tolerance. This can take a huge number of iterations, in typical cases about 3000 to 5000 model years, each of which involves thousands of time steps (e.g. 2880 steps for a three-hour step-size). Thus the overall number of iterations may be in the range of 10^6 to 10^7. When aiming at parameter optimization or sensitivity studies, the spin-up process has to be repeated several times, and thus in these cases a reduction of the computational time of a single spin-up run is even more important.

There are several strategies to reduce this computational effort. The following ones are more or less independent from each other: one of them is of course parallelization, usually by domain decomposition methods. The second one is the usage of precomputed *transport matrices* (see Khatiwala, 2007) that represent the (possibly linearized) tracer transport scheme applied in an ocean model. Monthly averaged matrices for the explicit and the implicit parts of the ocean tracer transport operator are usually used. In the ecosystem spin-up, these "climatological" matrices are then interpolated accordingly in every time step. With this method, the transport part of the ecosystem model reduces to matrix-vector multiplications, whereas the biogeochemical source-minus-sink terms are evaluated separately. A third way to reduce computational effort is to replace the standard spin-up (which, in mathematically terms, is a fixed-point iteration) by variants of Newton's method, which have higher convergence rates.

In this work we start from an implementation of a spin-up that applies the first two strategies. In order to drive the biogeochemical tracers, the software handles transport matrices that are stored in a common sparse format. Moreover, it uses routines of the Portable, Extensible Toolkit for Scientific Computation (PETSc; Balay et al., 1997, 2012) library to perform matrix-vector multiplications in parallel. The main advantages of this toolkit is that all Message Passing Interface (MPI; Walker and Dongarra, 1996) calls are hidden in built-in functions, and that optimized functions for matrix-vector operations (and more) already exist. The resulting software can be coupled with a wide range of biogeochemical models, as long as they conform to a rather flexible and general interface.

The main focus of this work is to describe the necessary changes to the software to port it to GPU hardware and to determine the resulting speed-up. High-performance computing on GPU or other special, highly parallel hardware is becoming more and more attractive in climate and geophysical research as well (e.g. Hanappe et al., 2011; Horn, 2012). To our knowledge there is no publication about using GPUs for marine ecosystem simulations. Since sparse matrix-vector multiplication (SpMVM) is an integral part of our spin-up implementation, this work is clearly motivated by the performance gains (up to a speedup of 24) achieved by the algorithms presented by Bell and Garland (2008). Moreover, we are interested in the behavior of the incorporated biogeochemical models ported to the GPU. For this purpose, we take here two examples with two tracers each. One of them is a simple linear model, describing for example the radioactive decay of two compounds. The second one is a well-known biogeochemical model that serves as a basis for more complex descriptions of the interplay of ocean biota and its major nutrients. It was used for numerical experiments by Parekh et al. (2005) or Kriest et al. (2010) for example.

Since we want to explicitly show what steps were necessary for the mentioned CPU-to-GPU port, we start by describing the original software for the ecosystem spin-up and the used biogeochemical models in Sect. 2. Afterwards we describe the standards, tools and libraries used for GPU programming in Sect. 3. We then show which GPU-adapted software can be used and what kind of adaption we additionally had to make in Sect. 4. We then show numerical results in Sect. 5 for the two models, both on CPU and GPU hardware. Finally, we conclude our work and give an outlook in Sect. 6.

2 Coupled marine tracer transport simulation using transport matrices

A marine ecosystem is usually modeled as a system of equations for the ocean circulation and the transport of temperature, salinity and the incorporated biogeochemical tracers, including their interactions. A fully coupled simulation – reflecting the fact that tracers are advected by the ocean circulation, their diffusion is dominated by the turbulent mixing of marine water, and, vice versa, a tracer concentration may effect the ocean circulation – is computationally expensive. Even on high-performance hardware, such a coupled (also called "online") simulation in three spatial dimensions is restricted to single model evaluations only, especially if steady annual cycles, which require long term spin-ups, are under investigation.

In contrast, a so-called "offline" computation is a simplified approach for tracers that are (or are regarded as) "passive", i.e. they do not affect the ocean physics, or this influence is neglected. This results in a one-way coupling from the ocean circulation to the tracer dynamics only, where the pre-computed circulation data (advection velocity vector field $\boldsymbol{v}$, mixing coefficient κ, temperature, and optionally salinity) enter the tracer transport equations as forcing.

With this data given, a marine ecosystem model considered in an offline computation consists of the following system of parabolic partial differential equations (here for n tracers y_i summarized in the vector $\boldsymbol{y} = (y_i)_{i=1,\dots,n}$):

$$\frac{\partial y_i}{\partial t} = \nabla \cdot (\kappa \nabla y_i) - \nabla \cdot (\boldsymbol{v}\, y_i) + q_i(\boldsymbol{y}), \quad i = 1, \dots, n, \qquad (1)$$

in the space–time cylinder $\Omega \times [0, T]$ with $\Omega \in \mathbb{R}^3$ being the spatial domain (i.e. the ocean) and $[0, T], T > 0$, the time interval. Here, we neglect the additional dependency on the space and time coordinates $(\boldsymbol{x}, t)$ in the notation for brevity. Additionally, homogeneous Neumann boundary conditions on $\Gamma = \partial\Omega$ for all tracers y_i are imposed. The source-minus-sink or coupling terms q_i in general are nonlinear and represent growth, dying, and tracer interaction. Each of them need not necessarily depend on *all* tracers in $\boldsymbol{y}$, but usually on more than the y_i itself. The q_i also include model parameters (as growth and dying rates, sinking velocities etc.) that are often subject to identification or estimation. They are usually spatially and temporally constant and not mentioned explicitly here.

2.1 Transport matrices

Since in an offline simulation the ocean circulation data is only used as pre-computed input for the tracer transport equations (Eq. 1), the spatial differential operators therein can be represented as a linear operator and the equations can be formally written as

$$\frac{\partial y_i}{\partial t} = L(\kappa, \boldsymbol{v}, t)\, y_i + q_i(\boldsymbol{y}), \quad i = 1, \dots, n. \qquad (2)$$

Here, $L(\kappa, \boldsymbol{v}, t)$ is a linear operator comprising the whole transport, i.e. diffusion and advection, for the given ocean circulation data κ and $\boldsymbol{v}$. It is time-dependent since the circulation data also depend on time, both in case of a transient simulation, and where a steady annual cycle driven by climatological data is sought. The operator L is identical for all tracers if the molecular diffusion of the tracers is small

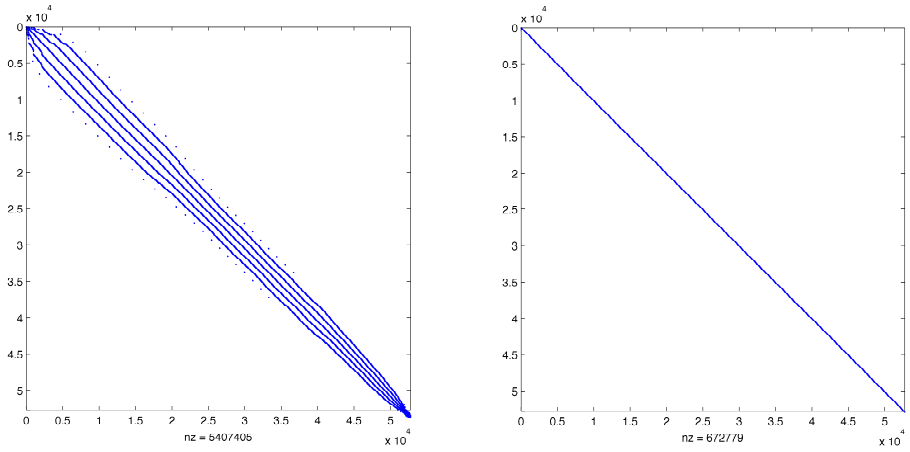


Fig. 1. One block of the explicit (left) and implicit (right) transport matrices $\mathbf{A}_{\text{exp}}, \mathbf{A}_{\text{imp}}$ computed using the MITgcm for a 2.8125° resolution (output of MATLAB®'s `spy` command).

compared to the turbulent mixing, which is a reasonable simplification.

The idea of the Transport Matrix Method (TMM) introduced in Khatiwala et al. (2005) is to compute or approximate the matrices that represent an appropriate discretization of L. This is done by running time steps of the ocean model that has produced the circulation data $\boldsymbol{v}, \kappa$ etc., with special, only locally non-zero initial distributions for one tracer. By varying the support of the initial distributions over the whole spatial domain, an approximation for one or several time steps can be obtained, which can be then used to build up a matrix representation of L. A comprehensive discussion of the temporal and spatial discretization as well as the process of evaluating transport matrices, especially in combination with operator splitting schemes can be found in Khatiwala et al. (2005). For our results we used twelve implicit and twelve explicit transport matrices, which represent monthly averaged diffusion and advection. The matrices are interpolated linearly to the corresponding discrete time step during simulation.

As a result, we obtain the following fully (temporal and spatial) discrete scheme where we now denote by $\boldsymbol{y}_j$ the appropriately arranged vector of the values of all n tracers on all spatial grid points at time step j. In the same way, we denote by $\boldsymbol{q}_j$ the vector of the discretized source-minus-sink terms at all spatial grid points in time step j. Using the TMM with a fixed time step-size τ, the time integration scheme for (Eq. 2) reads

$$\boldsymbol{y}_{j+1} = \mathbf{A}_{\text{imp},j}(\mathbf{A}_{\text{exp},j}\,\boldsymbol{y}_j + \tau\,\boldsymbol{q}_j(\boldsymbol{y}_j)) =: \varphi_j(\boldsymbol{y}_j). \tag{3}$$

Here n_τ is the total number of time steps and $\mathbf{A}_{\text{imp},j}, \mathbf{A}_{\text{exp},j}$ are the implicit and explicit transport matrices at time step $j = 0, \ldots, n_\tau - 1$. The matrices are block-diagonal and sparse and depend on the used time-stepping scheme: if – as a simple and unrealistic example – the whole system were solved explicitly by an Euler step, $\mathbf{A}_{\text{imp},j}$ would be the identity and $\mathbf{A}_{\text{exp},j}$ would be the discrete counterpart of $I + \tau L(\kappa, \boldsymbol{v}, t_j)$. Summarizing, starting from a vector $\boldsymbol{y}_0$ of initial values, each step in the time integration scheme (Eq. 3) to solve the tracer transport equations (Eq. 1) consists of the evaluation of the source-minus-sink term and two matrix-vector multiplications per tracer.

Table 1 shows typical values for the sizes and sparsity of transport matrices generated by the MIT General Circulation Model (MITgcm; Marshall et al., 1997) for two spatial resolutions, see Khatiwala et al. (2005); Piwonski and Slawig (2012). Since we deal with quadratic matrices and the sparsity patterns remain the same throughout the whole spin-up process a characterization of the used matrices by the number of rows (`nrows`) and the number of non-zero elements (`nnz`) is sufficient for our purpose. Figure 1 shows the sparsity patterns. The matrix entries are stored as *double precision* values.

2.2 Computation of steady annual cycles

Computing a periodic solution of the discretized system (Eq. 3) means looking for a fixed point of the mapping $\Phi = \varphi_{n_\tau - 1} \circ \cdots \circ \varphi_0$, i.e. for a trajectory $(\boldsymbol{y}_j)_{j=0,\ldots,n_\tau}$ with

$$\boldsymbol{y}_{n_\tau} = \Phi(\boldsymbol{y}_0) = \boldsymbol{y}_0. \tag{4}$$

Thus one application of the mapping Φ corresponds to the computation of one year model time (or model year). The time step used in our computations was 3 h, which corresponds (taking 360 days a year) to $n_\tau = 2880$. The discretization of the biogeochemical terms q_i may include shorter time steps (typically 8 per outer 3-h step).

The whole iteration to compute a steady cycle (or fixed point) now consists of a repeated application of the mapping Φ:

$$\boldsymbol{y}^{l+1} = \Phi(\boldsymbol{y}^l), \quad l = 0, \ldots, n_l - 1, \tag{5}$$

Table 1. Resolution, sizes and sparsity of one block of the explicit and implicit transport matrices for two resolutions computed with the MITgcm.

Horizontal resolution	Vertical layers	Matrix size (nrows)	Number of non-zeros, total (nnz) and percent	
			$\mathbf{A}_{\text{exp}}$	$\mathbf{A}_{\text{imp}}$
2.8125°	15	52 749	5 407 405 (0.1943 %)	672 779 (0.0024 %)
1°	23	682 604	76 567 216 (0.0164 %)	13 339 210 (0.0029 %)

where $\boldsymbol{y}^l$ is the vector of discretized tracer after l model years, i.e. $\boldsymbol{y}^l = \boldsymbol{y}_{l \cdot n_\tau}$, and n_l the total number of model years necessary to reach a steady annual cycle. The resulting structure of the spin-up is sketched in Algorithm 1.

From several computations it can be observed that after about $n_l = 3000$ iterations, a numerical steady solution (up to an accuracy of about 10^{-2} in discrete $L^2(\Omega)^n$ norm) is obtained. Thus we refer to this as a "converged steady annual cycle". This value of n_l was also used in (Kriest et al., 2010). The residual can be further decreased by using a higher number n_l of model years.

2.3 Applying parallel algorithms using the PETSc library

Obviously, a parallelization of the matrix-vector multiplication occurring every time step can significantly speed up the process of computing the steady annual cycle by the pseudo-time stepping (or fixed point iteration) described above. In the CPU setting (e.g. Piwonski and Slawig, 2012) the parallelization is carried out on a multi-processor, distributed-memory architecture. In order to avoid the direct implementation of MPI directives, we make use of the PETSc library. It is a collection of data structures and algorithms for the parallel solution of numerical problems and provides interfaces (APIs) to programming languages as Fortran, C, C++, Python, and MATLAB®. Main advantages of PETSc for our application are the parallelized matrix-vector-multiplication routines and the usage of an efficient sparse matrix storage format, in our case the default PETSc format, namely the "AIJ" or "Yale sparse" or "CSR" (compressed sparse row) format.

In our original implementation, the biogeochemical part (Algorithm 1, line 4) is implemented in Fortran, whereas the remainder of the code is realized in C. There is a difference with respect to the access of the tracer data that becomes important later on the GPU: for the biogeochemical computations (line 4), the values of the separate tracers and also on different spatial grid points (compare Eq. 6) are needed simultaneously. In contrast, the matrix-vector products (lines 6, 7) are executed separately for each tracer, thus allowing us to store and work with one block of the transport matrices only. Each matrix-vector product is computed by one call to the PETSc routine `MatMult()`.

For the interpolation step in line 5, three other PETSc routines are used (for explicit and implicit matrix separately) to compute the appropriately weighted matrices:

```
MatCopy(A[i_alpha], A_work, ...);
MatScale(A_work, alpha);
MatAXPY(A_work, beta, A[i_beta], ...);
```

These three routines together compute a linear interpolant or convex combination of two succeeding monthly averaged matrices, which are stored in the array `A` starting at index `i_alpha` and `i_beta`, respectively. Thus the above lines compute `A_work = alpha * A[i_alpha] + beta * A[i_beta]`, which gives the desired interpolated matrix in `A_work`, if `alpha, i_alpha` and `beta, i_beta` are chosen correctly with respect to the time step j.

2.4 Ecosystem and biogeochemical model examples

We use two simple models to test the computational gain possible with the GPU hardware. Each of them has two tracers (i.e. $n = 2$ in Eq. 1 and thereafter). Source codes for both models are available at Piwonski and Slawig (2012).

The first one is a simple radioactive decay model which is uncoupled and has the autonomous source-minus-sink term

$$\boldsymbol{q}(\boldsymbol{y}) = \begin{pmatrix} -\lambda_1\, y_1 \\ -\lambda_2\, y_2 \end{pmatrix}.$$

The parameters $\lambda_1, \lambda_2 > 0$ are the decay rates of the two radioactive elements. We chose Iodine I^{131} with $\lambda_1 \approx 44.88$ and Caesium Cs^{137} with $\lambda_2 \approx 0.0331$. This uncoupled model is used in order to test the gain in CPU time for the pure matrix-vector multiplication and interpolation in the TMM.

The second model is a typical biogeochemical model, including both coupling and nonlinearities. It is based on the N-DOP model described in Parekh et al. (2005), which was also used in Kriest et al. (2010), from which we basically take the notation. The model incorporates phosphate (nutrients, N, y_1) and dissolved organic phosphorus (DOP, y_2). The source-minus-sink term is split up into the upper, sun-lit or productive euphotic zone Ω_1 with depth z', and the lower, aphotic zone Ω_2:

Algorithm 1: Marine ecosystem spin-up using TMM

Require: Set of monthly averaged transport matrices $\mathbf{A}_{\text{imp}}, \mathbf{A}_{\text{exp}}$, initial tracer distribution $\boldsymbol{y}_0$, time step τ
Ensure : At the end $\boldsymbol{y}$ is a tracer distribution (at one point in time) of a steady annual cycle

$\boldsymbol{y} = \boldsymbol{y}_0$
repeat
  for $j = 0, \ldots, n_\tau - 1$ **do**
    compute biogeochemical source-minus-sink terms: $\tilde{\boldsymbol{y}} = \boldsymbol{q}_j(\boldsymbol{y})$
    interpolate the monthly averaged transport matrices to the current time step j
    perform explicit step: $\hat{\boldsymbol{y}} = \mathbf{A}_{\text{exp},j}\,\boldsymbol{y}$
    perform implicit step: $\boldsymbol{y} = \mathbf{A}_{\text{imp},j}(\hat{\boldsymbol{y}} + \tau\,\tilde{\boldsymbol{y}})$
  end
until *steady annual cycle is reached*

$$q_1(\boldsymbol{y}) = \begin{cases} -f(y_1) + \lambda\ y_2 & \text{in } \Omega_1 \\ (1-\sigma)\,\frac{\partial}{\partial z} F(y_1) + \lambda\ y_2 & \text{in } \Omega_2 \end{cases}$$

$$q_2(\boldsymbol{y}) = \begin{cases} \sigma f(y_1) - \lambda\ y_2 & \text{in } \Omega_1 \\ -\lambda\ y_2 & \text{in } \Omega_2 \end{cases}$$

z being the vertical coordinate. The biological production is calculated as a function

$$f(y_1) = \alpha\ \frac{y_1}{y_1 + K_{\text{N}}}\ \frac{I}{I + K_{\text{I}}}$$

of nutrients y_1 and light I. The dependence on the latter is omitted here in the notation for brevity. The production is limited by a half saturation function, also known as Michaelis-Menten kinetics, and a maximum production rate parameter α. Light is modeled as a portion of shortwave radiation I_{SWR}, which is computed as a function of latitude and season following the astronomical formula of Paltridge and Platt (1976). The portion depends on the photo-synthetically available radiation $\sigma_{\text{PAR}} = 0.4$, the ice cover σ_{ice}, and the exponential attenuation of water, i.e.

$$I = I_{\text{SWR}}\ \sigma_{\text{PAR}}\ (1 - \sigma_{\text{ice}})\ \exp(-z\,K_{\text{H}_2\text{O}}).$$

A fraction σ of the biological production remains suspended in the water column as dissolved organic phosphorus, which remineralizes with a rate λ. The remainder of the production sinks as particulate to depth where it is remineralized according to the empirical power–law relationship determined by Martin et al. (1987),

$$F(y_1) = \left(\frac{z}{z'}\right)^{-b} \int_0^{z'} f(y_1)\,dz. \tag{6}$$

Similar modeling of biological production can be found for example in Dutkiewicz et al. (2005). Algorithm 2 sketches the implementation of the N-DOP model, whereas the model parameters are given in Table 2.

Table 2. Parameters in the N-DOP model.

Name	Description	Unit
λ	remineralization rate of DOP	d^{-1}
α	maximum community production rate	d^{-1}
σ	fraction of DOP	1
K_{N}	half saturation constant of N	mmol P m^{-3}
K_{I}	half saturation constant of light	W m^{-2}
$K_{\text{H}_2\text{O}}$	attenuation of water	m^{-1}
b	sinking velocity exponent	1

3 GPU computing with CUDA

In this section we describe the basic architecture of GPUs and give an overview of some useful libraries. We concentrate on NVIDIA's Compute Unified Device Architecture (CUDA; NVIDIA Corporation, 2012). One alternative is, for example, OpenCL (The Khronos Group, 2012).

NVIDIA, as one of the leading producers of graphic cards, has developed its own parallel architecture for executing computationally expensive code on GPUs. By exploiting the architecture of graphic cards as well as the increased memory bandwidth, it is possible to perform a far greater number of floating point operations per second (FLOPS) than on CPUs. While CPUs have about one to eight cores each with up to 4 GHz clock rate, GPUs nowadays do have a lower clock rate, but hundreds of cores which can run multiple threads simultaneously.

The basic unit of the CUDA programming model is called *kernel*. A kernel is a piece of program code invoked on the CPU *host* and executed on the GPU *device* by threads. These threads are organized in a "grid" of thread "blocks". A call to

```
kernel<<<gridSize, blockSize>>>();
```

creates `gridSize` blocks of `blockSize` threads ready for execution, whereas the order of processing the blocks depends on the hardware.

Algorithm 2: Computation of $\tilde{\boldsymbol{y}} = \boldsymbol{q}_j(\boldsymbol{y})$ for the N-DOP model.

Require: Tracer vectors $\boldsymbol{y}$, latitude ϕ, ice cover σ_{ice}, depths z, layer heights dz and parameters: $\lambda, \alpha, \sigma, K_{\text{N}}, K_{\text{H}_2\text{O}}, K_{\text{I}}, b$
Ensure : $\tilde{\boldsymbol{y}}$ consists of the computed sinks and sources

for *every water column i with* n_i *layers* **do**
  $I = 0.4 * (1 - \sigma_{\text{ice},i}) * I_{\text{SWR}}(\phi_i)$ // compute insolation
  $\tilde{\boldsymbol{y}} = 0$ // zero all bio steps
  for *8 biosteps* **do**
    $\boldsymbol{y}' = \boldsymbol{y} + \tilde{\boldsymbol{y}}$ // take previous steps into account
    $\tilde{\boldsymbol{y}}' = 0$ // zero one bio step
    for *layer* $j = 1$ *to* $\min(n_i, 2)$ **do** // production layers
      $I_j = I * \exp(-z_j \; K_{\text{H}_2\text{O}})$
      $f_j = \alpha * y'_{1,j} * I_j / (y'_{1,j} + K_{\text{N}}) / (I_j + K_{\text{I}})$
      $\tilde{y}'_{1,j} = \tilde{y}'_{1,j} - f_j$
      $\tilde{y}'_{2,j} = \tilde{y}'_{2,j} + \sigma * f_j$
      if *last layer* **then**
        $\tilde{y}'_{2,j} = \tilde{y}'_{2,j} + (1 - \sigma) * f_j$
      else
        for *every layer k beneath* **do** // approximation of dF/dz
          if *last layer* **then**
            $\tilde{y}'_{2,j} = \tilde{y}'_{2,j} + (1 - \sigma) * f_j * dz_j * (z_{k-1}/z_j)^{-b} / dz_k$
          else
            $\tilde{y}'_{2,j} = \tilde{y}'_{2,j} + (1 - \sigma) * f_j * dz_j * ((z_{k-1}/z_j)^{-b} - (z_k/z_j)^{-b}) / dz_k$
          end
        end
      end
    end
    for *layer* $j = 1$ *to* n_i **do** // all layers
      $\tilde{y}'_{1,j} = \tilde{y}'_{1,j} + \lambda * y'_{2,j}$
      $\tilde{y}'_{2,j} = \tilde{y}'_{2,j} - \lambda * y'_{2,j}$
    end
    $\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{y}} + 1/8 * \tilde{\boldsymbol{y}}'$ // scale and add to all bio steps
  end
end

The GPU hardware consists of several Streaming Multiprocessors (SMs). Each SM has its own buffer memory, registers, and a number of cores. The cores have their own units for integer and floating-point calculation. For example, the GeForce GTX 480 used here has 15 SMs with 32 cores each, i.e. a total of 480 cores. On a core, the smallest executable unit is a "warp", which consists of 32 threads. The total number of threads that can run simultaneously on a multiprocessor is dependent on the Compute Capability (CC) of the graphics chips. For the GTX 480 the limit is 1536 threads, which results in a maximum number of concurrent threads for the entire GPU of $15 \times 1536 = 23\,040$ (p. 159, NVIDIA Corporation, 2011).

The device memory on the GPU is divided into three types of physical and virtual portions. At first, a thread has access to its own private memory which is, depending on the CC, between 16 kB and 512 kB. Secondly, threads within one block have access to a shared memory of between 16 kB and 48 kB. Finally, all threads have access to a shared global memory whose size is limited by the total amount of memory of the GPU. In order to run kernel code on the GPU, all data must be transferred from the host memory of the CPU to the device memory on the GPU.

NVIDIA provides a compiler (`nvcc`) that translates C code into the CUDA Instruction Set (called PTX) and behaves similarly to the C compiler (`gcc`) included in the GNU Compiler Collection (GCC). A port of the GNU debugger `gdb` is also included in the CUDA toolkit.

3.1 Libraries

We make use of libraries that provide basic algorithms while working with GPUs. The first one is: Thrust (Bell and Hoberock, 2011), a C++ collection of generic algorithms, similar to the C++ Standard Template Library (STL), that exploit the parallelism of the GPU in a transparent way. Using Thrust, many problems can be solved without even writing code for

the GPU. For documentation and sample code we refer to (Hoberock and Bell, 2012).

The second library is Cusp (Bell and Garland, 2010), which provides data types for sparse matrices and algorithms for basic linear algebra operations on them. All data structures in Cusp have a parameter that determines whether it is stored in CPU or GPU memory. Operations on the data will then take place in the respective storage area. For our application, in particular the structure `cusp::csr_matrix` for the CSR format and the matrix-vector multiplication routine `cusp::multiply` that uses the algorithm described in Bell and Garland (2008, 2009), which was specially developed for GPUs, are important. Documentation and sample code can be found at (Bell and Garland, 2010).

The third library we used was the preliminary implementation of PETSc for the CUDA architecture presented in Minden et al. (2010). With the help of the Thrust and Cusp libraries, a large part of the PETSc `Vector` and some parts of the `Matrix` class have been implemented. The fundamental problems of interaction of PETSc with the GPU have been resolved, but only the routines that were necessary for the example treated in Minden et al. (2010) have been implemented. Basically this "PETSc GPU" extends the built-in structures by a value that indicates in which memory the most recent data are stored. This guarantees that the correct data is available (and if necessary copied to) the memory that is currently used. Here, we employed the developer PETSc library version 3.2-p5.

4 Port of the marine ecosystem simulation onto the GPU

We now describe the necessary modifications and extensions of the original program that was running on a multi-processor CPU cluster in order to perform the simulation on a GPU. Basically these modifications are extensions of PETSc GPU, modifications necessary to use the CUDA Fortran compiler for the biogeochemical model code and some routines for conversions between different data alignments.

4.1 Necessary extensions of PETSc GPU

The preliminary PETSc GPU implementation was designed to solve systems of equations, and thus not all functions necessary for our applications were included. To avoid any copying of data between CPU and GPU storage that would have destroyed the speed-up, we had to extend the library. In our case, the three PETSc routines `MatCopy`, `MatScale`, and `MatAXPY` mentioned in Sect. 2.2 had to be modified.

If using sparse matrices with PETSc and working with GPUs the PETSc wrapper function

```
MatCopy(Ain, Aout, ...);
```

accesses `MatCopy_SeqAIJCUSP()` to copy the values from matrix `Ain` to `Aout`. Here, it is theoretically possible that both matrices are either currently in the GPU memory, the CPU memory or in both. For a complete and correct implementation, it would have been necessary to cover all these cases, and accordingly select the memory the matrices are actually copied to. For our application it was sufficient to cover only the case where the matrices are both in the GPU memory, thus only this case was implemented. Therefore, an additional call to `MatCUSPCopyToGPU()` in `MatCopy_SeqAIJCUSP()` ensures that both matrices are in the GPU memory.

The PETSc routines `MatScale()` and `MatAXPY()` implement typical linear algebra subproblems, which are only performed on the non-zero matrix elements. Consequently, they could be completely realized using the Cusp BLAS library for the GPU.

4.2 PGI CUDA-Fortran

Many biogeochemical models are implemented in Fortran. Mostly, they are part of a software that has evolved over decades (e.g. MITgcm). Since we want to use them with GPUs without any modification to original source code, we need a Fortran compiler and the appropriate libraries. At the time of this work there was only one candidate, namely the PGI CUDA Fortran compiler (The Portland Group, 2012). It extends the language by constructs for calling kernel as well as the CUDA API functions. The syntax of a kernel call in Fortran is

```
call kernel<<<gridSize, blockSize>>>()
```

and thus similar to CUDA C++ . There are some extensions compared to CUDA C++, but also some restrictions. For details we refer to the manual (The Portland Group, 2011a, p. 14).

4.3 Other extensions to the implementation on the CPU

As mentioned in Sect. 2.3, there are two different data alignments useful for the spin-up using the TMM: one for the biogeochemical source-minus sink terms, where all tracers of a water column are kept in a contiguous piece of memory, and another one for the multiplication with the transport matrices, where every water column of a tracer is kept together to reduce the storage requirements for the matrices. Thus a copying between these two data alignments is necessary in every step of the algorithm. For the use on the GPU, three copying functions in the original code were additionally modified using the Thrust library.

4.4 The compilation process for the GPU

Here we briefly sketch the overall compilation and linking process of the resulting code for the use on the GPU. The process is visualized in Fig. 2.

In a first step (top right in Fig. 2) the biogeochemical model file `model.F` is included into `driver.CUF`

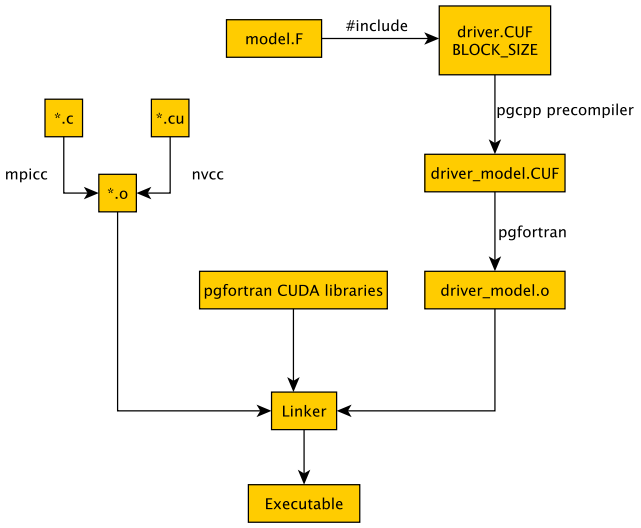


Fig. 2. Compilation and linking process of the spin-up for usage on the GPU.

and processed to `driver_model.CUF` by the preprocessor of the C++ compiler `pgcpp`. The Fortran compiler `pgfortran` then generates the object file `driver_model.o`.

The driver routine `driver.CUF` has two tasks: at first the Fortran compiler requires that all functions which shall run on the GPU are marked with the `device` attribute, see The Portland Group (2011b). Since the compiler has no ability to set default attributes for all functions, it is necessary to integrate them through a preprocessor macro. Therein the Fortran keyword `subroutine` is replaced by `attributes(device) subroutine`. Secondly, the driver provides support functions for the three entry points into the biogeochemical model, namely (i) the evaluation of the source-minus-sink term, (ii) the initialization and (iii) deinitialization of the model. These three functions need corresponding kernels for the GPU. This approach ensures the original Fortran interface of the biogeochemical model remains unaltered.

In a second step (top left of Fig. 2) the original, unmodified C code is compiled with the MPI wrapper of the GNU C compiler `mpicc`, while CUDA extensions are translated with `nvcc`. Finally all object code files are linked against PGI Fortran libraries, which results in the final executable.

5 Numerical results

In this section we compare the performance of the spin-up on our CPU/GPU test hardware. We use the two models described in Sect. 2.4. A special emphasis lies on the time needed for the individual parts, namely the evaluation of the biogeochemical source minus-sink term, the matrix interpolation and the matrix-vector multiplication. Moreover, we contrast the best GPU result for the N-DOP model with results from three different distributed-memory architectures.

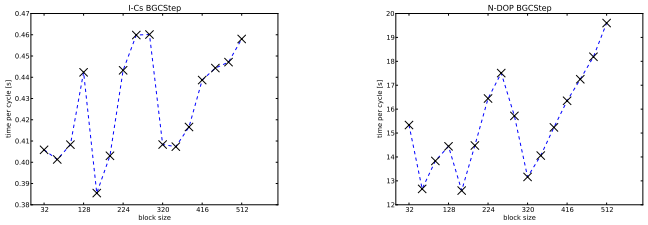


Fig. 3. Computational time needed for the I-Cs (left) and N-DOP (right) model within one model year depending on the block size.

5.1 Setup

The CPU/GPU test hardware consists of two GeForce GTX 480 graphic cards and two Intel® Xeon® E5520 CPUs running at 2.27 GHz. However, the following tests were performed only on *one* GPU and only on *one core* of the CPU. No display was connected to the graphic card and computations on the GPU were performed with *double precision*, which is natively supported by the GTX 480. The theoretical peak performance of the GPU is at 168 GFlop s^{-1} and the internal bandwidth at 177 GB s^{-1}. The performance of one core of the CPU system is at 9.08 GFlop s^{-1}, its bandwidth at 21.2 GB s^{-1}.

To test a specific biogeochemical model, the software is compiled with the according source code and run for 100 model years. In detail, when the executable starts the data (matrices, initial vectors, etc.) is copied into the CPU or GPU memory and 100 iterations, 2880 time steps each, are performed consecutively. In the case of a GPU run, the results are copied back to CPU memory at the end.

Thus, the whole data has to fit into the memory of the device (or host). This is the case if the 2.8125° horizontal resolution is used. Here, the 1.5 GB RAM of a GTX 480 (or 40 GB of the CPU system) are enough for about 1 GB of data. However, a monthly averaged set of transport matrices based on a 1° resolution (approximately 13 GB) is too large for the used GPU system. Such an amount of data requires a different approach (see Sect. 6). Hence, we focus on the 2.8125° resolution and omit profiling of data transfers between CPU and GPU memory.

When processing source codes, the `mpicc`, `mpif90` and `nvcc` compilers are switched to `-O` (i.e. optimize). For `pgfortran` no optimization flags are used. To perform time measurements, the profiling system of PETSc is applied. No further source code optimization is performed regarding the GPU.

Table 3. Minimum, maximum, average and standard deviation of computational time for one model year spent on the CPU and GPU. Shown are results of 100 model years, each year timed separately. `BGCStep` block size: 160.

	CPU				GPU				
Model	Min	Max	Avg	StdDev	Min	Max	Avg	StdDev	CPU Min : GPU Min
I-Cs	159.58 s	161.44 s	160.19 s	0.47	15.49 s	15.52 s	15.50 s	0.002	10.30
N-DOP	621.43 s	626.79 s	622.14 s	0.54	28.17 s	28.20 s	28.18 s	0.003	22.06

Table 4. The three main portions in every time step of the spin-up.

Lines in Alg. 1	Routine	Description
4	`BGCStep`	Evaluation of source-minus-sink terms
5	`MatCopy`, `MatScale`, `MatAXPY`	Interpolation of transport matrices
6, 7	`MatMult`	Multiplication of transport matrices with tracer vectors

5.2 Results

We start by examining the block size parameter for the Fortran kernel calls of the biogeochemical model. The block size describes the number of vertical profiles (or water columns) that are processed within a block. While the grid and block dimensions are calculated automatically, if using Thrust or Cusp for example, a suitable value for the Fortran kernel must be determined experimentally for the time being. For all tests we use just 100 model years (instead of 3000 or more needed in practice, see Sect. 2.2) to render the numerical experiments feasible, especially when simulating the N-DOP model on the CPU, which still takes about 17 h.

Figure 3 depicts the mean of 100 model years' computational time spent on the GPU for biogeochemical model steps depending on the block size. In both models, strong fluctuations up to 100 % occur. However, both graphs show similar occurrence of minima and maxima. We suppose this is due to the unbalanced distribution of water columns (see Sect. 6). However, the absolute minimum (I-Cs: 0.38 s, N-DOP: 12.6 s) is obtained for a block size of 160.

This value is used for the subsequent test, in which every year is timed separately. Table 3 shows the minimum, maximum and mean of computational time for one model year spent on the CPU and GPU. The standard deviation is small on the CPU (I-Cs: 0.47, N-DOP: 0.54) and marginal on the GPU (I-Cs: 0.002, N-DOP: 0.003). However, the overall reduction is about 10 for the simpler I-Cs model and about 22 for the more complex N-DOP model, a difference we investigate further.

Thus, the next tests focus on the individual steps within the repeat-until loop of Algorithm 1, corresponding to one annual cycle. The invoked routines are listed in Table 4. Their individual performance gain is depicted in Table 5. Regarding `MatCopy`, `MatScale`, `MatAXPY` and `MatMult`, we see a similar relative performance gain for both models from about 9 to 13. In contrast, `BGCStep` shows a speed-up of about 10 for the I-Cs model, whereas for N-DOP a ratio between the CPU and GPU of 36 can be observed. In addition, in Fig. 4 we recognize that 75 % of the overall time on the CPU, which is spent for the evaluation of the N-DOP model, is sped up by this factor on the GPU. This explains the overall ratio of 22. Note that the slightly higher average computational times in Fig. 4 (compared to those in Table 3) are due to the higher granularity of profiling. Moreover, we see that the computational effort for the I-Cs model, which is just a scaling of the tracer vector, is smaller than 3 % on both architectures. Here, the overall speed up is dominated by the matrix operations.

Concerning the latter, we pick `MatMult` for a detailed view on performance and bandwidth and compare our results with those reported by Bell and Garland (2008). We calculate the number of floating point operations for one model year as follows:

$$n_{\text{ops}} = n_\tau * 2 * (2 * \mathtt{nnz}_{\text{exp}} + 2 * \mathtt{nnz}_{\text{imp}}) \approx 70\ \text{GFlop}\,,$$

which is the number of time steps per year *times* number of tracers *times* (explicit plus implicit) sparse matrix-vector multiplication, which is exactly twice the number of nonzeros. We consider the results from the N-DOP model and divide n_{ops} by 58.19 s (CPU) and 5.87 s (GPU), respectively. We obtain a performance of approximately $1.2\,\text{GFlop}\,\text{s}^{-1}$ for the CPU and $11.9\,\text{GFlop}\,\text{s}^{-1}$ for the GPU. This is about 13 % of the theoretical peak performance of one CPU core ($9.08\,\text{GFlop}\,\text{s}^{-1}$) and about 7 % of $168\,\text{GFlop}\,\text{s}^{-1}$, regarding the GPU. The poor performance is due to the bandwidth limitation, which is typical for sparse matrix-vector multiplications. Following Bell and Garland (2008), we multiply n_{ops} by $10\,\text{Byte}\,\text{Flop}^{-1}$ (CSR vector kernel) and relate the result to the computational time spent on the CPU and GPU, respectively. We obtain 56.8 % ($12\,\text{GB}\,\text{s}^{-1}$) of the theoretical bandwidth for the CPU and 67.4 % ($119.4\,\text{GB}\,\text{s}^{-1}$) for the GPU.

Table 5. Mean computational time within one model year and performance gains of the individual routines depicted in Table 4.

	I-Cs			N-DOP		
Routine	CPU	GPU	CPU : GPU	CPU	GPU	CPU : GPU
`BGCStep`	3.79 s	0.38 s	9.93	469.76 s	13.05 s	36.00
`MatCopy`	34.52 s	3.91 s	8.83	34.04 s	3.91 s	8.70
`MatScale`	23.83 s	1.99 s	11.96	23.33 s	1.99 s	11.70
`MatAXPY`	38.71 s	2.89 s	13.39	37.49 s	2.89 s	12.96
`MatMult`	60.04 s	5.87 s	10.22	58.19 s	5.87 s	9.92

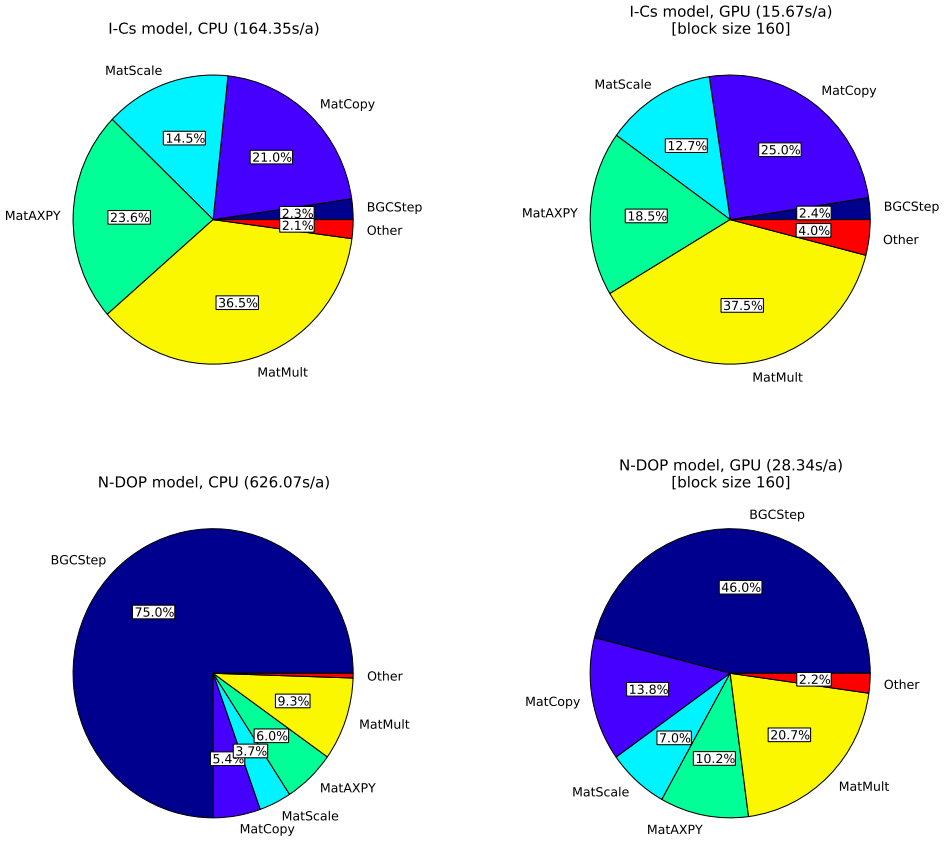


Fig. 4. Fraction of computational time needed for the individual parts in one year of the spin-up (Algorithm 1 and Table 4) for the I-Cs (top) and the N-DOP (bottom) model on the CPU (left) and GPU (right).

These figures in turn are satisfying and confirm a good performance of the CSR vector kernel used by `MatMult`. However, they also show that a sparse matrix-vector multiplication on a GTX 480, which is two generations ahead of the GTX 280 used by Bell and Garland (2008), is only slightly faster. Here, we refer to the 10 GFlop s^{-1}, achieved by the GTX 280 for "unstructured" matrices, compared to the 11.9 GFlop s^{-1} achieved by the GTX 480 for the transport matrices. This is obviously due to the only slightly increased memory bandwidth from 141.7 GB s^{-1} (GTX 280) to 177 GB s^{-1} (GTX 480).

Nevertheless, motivated by the overall speed up, we perform simulations of the N-DOP model on three different CPU clusters and put them in relation to the best performance on the GPU as a last comparison. Figure 5 shows that a GTX 480 can compete with approximately 56 Barcelona, 28 Westmere, and 17 Gainestown processors.

6 Conclusions

In order to port our existing implementation of the spin-up of marine ecosystem models using transport matrices from CPU to GPU hardware, modifications of our own code and

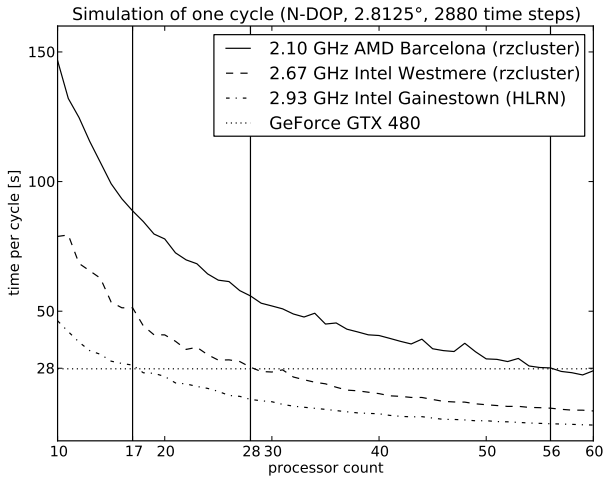


Fig. 5. Comparison between CPU cluster and the used GPU for one model year for the N-DOP model, ("rzcluster" refers to the Kiel University cluster, "HLRN" to the cluster of the *North-German Supercomputing Alliance*).

extensions to the used libraries were necessary. This work required knowledge in the computing architecture of the used CUDA programming framework and the PETSc, Thrust and Cusp libraries. In order to compile Fortran code for the GPU, a commercial compiler was necessary.

Concerning the computational gain of the used biogeochemical models, we were surprised by the good performance of the N-DOP implementation. Here, we can only speculate about the reasons and see a need for a more detailed investigation. Considering the complexity of Algorithm 2, however, such an effort was out of the scope of this work. We thus reported only results here.

Regarding `MatMult`, we observed a similar good utilization of memory bandwidth by the CSR vector kernel for transport matrices as reported by Bell and Garland (2008) for "unstructured" matrices. Moreover, all matrix operations showed a satisfactory performance gain.

Our results motivate us to investigate other biogeochemical models and to get to the bottom of the significantly higher speed-up of the N-DOP model compared to other operations. Additionally, we are eager to prepare the code for usage with multiple GPUs and/or techniques of simultaneous copying and computing.

Acknowledgements. The GPU computations were performed with kind permission and support of the Research Group for Communication Systems at Christian-Albrechts-Universität of Kiel, Institute for Computer Science. The access to the HLRN computing facility was kindly provided by the group Marine Biogeochemical Modelling at IfM GEOMAR, Kiel, Germany. Moreover, the authors would like to thank two anonymous reviewers for their very helpful comments.

Edited by: D. Ham

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., Birkhäuser Press, 163–202, 1997.

Balay, S., Buschelman, K., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Smith, B. F., and Zhang, H.: PETSc Web page, available at: http://www.mcs.anl.gov/petsc (last access: 7 January 2013), 2012.

Bell, N. and Garland, M.: Efficient sparse matrix-vector multiplication on CUDA, Technical Report NVR-2008-04, NVIDIA Corporation, 2008.

Bell, N. and Garland, M.: Implementing sparse matrix-vector multiplication on throughput-oriented processors, in: SC '09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, New York 2009, ACM, 1–11, 2009.

Bell, N. and Garland, M.: Cusp: Generic Parallel Algorithms for Sparse Matrix and Graph Computations, available at: http://cusp-library.googlecode.com (last access: 7 January 2013), 2010.

Bell, N. and Hoberock, J.: Thrust: A Producivity-Oriented Library for CUDA, GPU Computing Gems Jade Edition, 2011.

Dutkiewicz, S., Follows, M., and Parekh, P.: Interactions of the iron and phosphorus cycles: A three-dimensional model study, Global Biogeochem. Cy., 19, 1–22, 2005.

Hanappe, P., Beurivé, A., Laguzet, F., Steels, L., Bellouin, N., Boucher, O., Yamazaki, Y. H., Aina, T., and Allen, M.: FAMOUS, faster: using parallel computing techniques to accelerate the FAMOUS/HadCM3 climate model with a focus on the radiative transfer algorithm, Geosci. Model Dev., 4, 835–844, doi:10.5194/gmd-4-835-2011, 2011.

Hoberock, J. and Bell, N.: Thrust, available at: http://thrust.github.com (last access: 7 January 2013), 2012.

Horn, S.: ASAMgpu V1.0 – a moist fully compressible atmospheric model using graphics processing units (GPUs), Geosci. Model Dev., 5, 345–353, doi:10.5194/gmd-5-345-2012, 2012.

Khatiwala, S.: A computational framework for simulation of biogeochemical tracers in the ocean, Global Biogeochem. Cy., 21, GB3001, doi:10.1029/2007GB002923, 2007.

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

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

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

Martin, J. H., Knauer, G. A., Karl, D. M., and Broenkow, W. W.: VERTEX: carbon cycling in the northeast Pacific, Deep Sea Res. Part A, 34, 267–285, 1987.

Minden, V., Smith, B., and Knepley, M.: Preliminary implementation of PETSc using GPUs, in: Proceedings of the 2010 International Workshop of GPU Solutions to Multiscale Problems in Science and Engineering, Harbin Nov. 2010, Springer, 2010.

NVIDIA Corporation: CUDA C Programming Guide, available at: http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf (last access: 7 January 2013), 2011.

NVIDIA Corporation: CUDA, available at: http://www.nvidia.com/object/cuda_home_new.html (last access: 7 January 2013), 2012.

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

Parekh, P., Follows, M. J., and Boyle, E. A.: Decoupling iron and phosphate in the global ocean, Global Biogeochem. Cy., 19, GB2020, doi:10.1029/2004GB002280, 2005.

Piwonski, J. and Slawig, T.: Metos3D: A Marine Ecosystem Toolkit for Optimization and Simulation, CAU Kiel, Institut für Informatik, available at: https://github.com/metos3d (last access: 7 January 2013), 2012.

The Khronos Group: OpenCL – The open standard for parallel programming of heterogeneous systems, available at: http://www.khronos.org/opencl/ (last access: 7 January 2013), 2012.

The Portland Group: PGI Fortran Compiler Reference Manual, available at: http://www.pgroup.com/doc/pgiref.pdf (last access: 7 January 2013), 2011a.

The Portland Group: CUDA Fortran Programming Guide and Reference, available at: http://www.pgroup.com/doc/pgicudafortug.pdf (last access: 7 January 2013), 2011b.

The Portland Group: PGI CUDA-Fortran Compiler, available at: http://www.pgroup.com/resources/cudafortran.htm (last access: 7 January 2013), 2012.

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

Ocean Modelling 68 (2013) 22–36

Contents lists available at SciVerse ScienceDirect

Ocean Modelling

journal homepage: www.elsevier.com/locate/ocemod

Accelerated parameter identification in a 3D marine biogeochemical model using surrogate-based optimization

M. Prieß [b,*], J. Piwonski [a], S. Koziel [c], A. Oschlies [b], T. Slawig [a]

[a] Christian-Albrechts Universität zu Kiel, Department of Computer Science, 24098 Kiel, Germany
[b] GEOMAR – Helmholtz Centre for Ocean Research Kiel, Düsternbrooker Weg 20, 24105 Kiel, Germany
[c] Engineering Optimization & Modeling Center, School of Science and Engineering, Reykjavik University, Menntavegur 1, 101 Reykjavik, Iceland

ARTICLE INFO

Article history:
Received 16 November 2012
Received in revised form 11 April 2013
Accepted 12 April 2013
Available online 3 May 2013

Keywords:
Marine biogeochemical model
Parameter identification
Model calibration
Surrogate-based optimization
Response correction
Low-fidelity model

ABSTRACT

We present the application of the Surrogate-based Optimization (SBO) method on a parameter identification problem for a 3-D biogeochemical model. SBO is a method for acceleration of optimization processes when the underlying model itself is of very high computational complexity. In these cases, coupled simulation runs require large amounts of computer time, where optimization runs may become unfeasible even with high-performance hardware. As a consequence, the key idea of SBO is to replace the original and computationally expensive (high-fidelity) model by a so-called surrogate, which is created from a less accurate but computationally cheaper (low-fidelity) model and a suitable correction approach to increase its accuracy. To date, the SBO approach has been widely and successfully used in engineering applications and also for parameter identification in a 1-D marine ecosystem model of NPZD type. In this paper, we apply the approach onto a two-component biogeochemical model. The model is spun-up into a steady seasonal cycle via the Transport Matrix Approach. The low-fidelity model we use consists of a reduced number of spin-up iterations (several decades instead of millennia used for the original model). A multiplicative correction operator is further exploited to extrapolate the rather inaccurate low-fidelity model onto the original one. This corrected model builds our surrogate. We validate this SBO method by twin-experiments that use synthetic observations generated by the original model. We motivate our choice of the low-fidelity model and the multiplicative correction and discuss the computational advantage of SBO in comparison to an expensive parameter optimization in the context of the high-fidelity model. The proposed SBO technique is shown to yield a solution close to the target at a significant gain of computational efficiency. Without further regularization techniques, the method is able to identify most model parameters. The method is simple to implement and presents a promising and pragmatic tool to calibrate biogeochemical models in a global three-dimensional setting.

1. Introduction

Determining the parameters of the source-minus-sink terms in a given marine ecosystem component of biogeochemical ocean models is a challenging task in ocean modeling, especially when the ecosystem model is embedded into a spatially three-dimensional ocean circulation. Since there is still no agreement upon what is the correct ecosystem model or model structure, a fair assessment of the different models depends on their validation against given observational data. This validation process intrinsically requires parameter optimization runs to estimate the model's capability in representing the data and thus to be appropriate and valid for prognostic simulations.

Solving this nonlinear optimization problem, whether deterministic (e.g., gradient-based) or stochastic (e.g., meta-heuristics) algorithms are used, typically requires a large number of evaluations of the model-data misfit function (sometimes called cost or objective function). For global models on larger spatial scales, this translates into prohibitively high computational costs since a single model evaluation is already computationally expensive (Oschlies, 2006). Straightforward attempts to employ the full model directly in conventional optimization algorithms are therefore generally tedious or even infeasible, even on high performance computers. As a consequence, a careful and systematic assessment of a model's ability to reproduce observed data is still the exception rather than the rule (Arhonditsis and Brett, 2004; Kwon and Primeau, 2006, 2008; Mattern et al., 2012). Methods that accelerate

* Corresponding author. Tel.: +49 (0) 431 600 4289; fax: +49 (0) 431 600 4469.
E-mail addresses: mpriess@geomar.de (M. Prieß), jpi@informatik.uni-kiel.de (J. Piwonski), koziel@ru.is (S. Koziel), aoschlies@geomar.de (A. Oschlies), ts@informatik.uni-kiel.de (T. Slawig).

1463-5003/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.ocemod.2013.04.003

either the optimization process itself or the underlying simulation are in high demand.

With respect to the latter, the *Transport Matrix Method (TMM)* introduced in Khatiwala et al. (2005) allows to efficiently compute the distribution of biogeochemical tracers for a given climatological ocean circulation. Pre-computed by an ocean model, the discretized diffusive and advective transport rates are stored in the form of linear transport matrices, which can then be used for *offline* transport computations of passive variables (i.e. not inducing dynamic effects) such as biogeochemical tracers insofar as their generally very small effects on density, e.g. via solar heating (Oschlies, 2004), can be neglected.

When applied to simulate a steady-state annual cycle in the marine ecosystem, the TMM can be efficiently used in a spin-up simulation (i.e., a pseudo-time stepping or fixed point iteration) either in a pure forward mode or using Newton's method for determining a periodic fix-point solution, as described for example in Khatiwala (2008). Using sophisticated numerical libraries and parallelization strategies, a flexible environment for simulation of a whole class of biogeochemical models has been developed (Piwonski and Slawig, 2010, 2011).

Attempts to seek a feasible optimization of global biogeochemical models include the work by Kwon and Primeau, (2006, 2008), and Mattern et al. (2012). Therein, two different approaches have been exploited for the parameter optimization, a statistical emulator technique as well as a gradient-free method, the Nelder–Mead simplex algorithm.

In this paper, we present another strategy called *Surrogate-based Optimization (SBO)*. This approach can be very powerful in terms of reducing the computational cost of a parameter optimization run when compared to rather standard optimization algorithms, as for example the one used in Kwon and Primeau (2006, 2008). SBO is widely used in engineering applications (Bandler et al., 2004; Leifsson and Koziel, 2010; Forrester and Keane, 2009; Queipo et al., 2005).

The principal idea of SBO is to replace the direct optimization of the original typically expensive *high-fidelity* model by iterative updating and re-optimization of its *surrogate*. The surrogate is a computationally cheap and yet reasonably accurate representation. A well performing SBO algorithm leads to a solution close to the one of the original optimization problem. Furthermore, the number of high-fidelity model evaluations in a SBO approach is typically significantly smaller when compared to a direct high-fidelity model optimization using conventional approaches. This can allow for significant reduction of the computational cost.

The surrogate can be created by different approaches: using approximations of sampled high-fidelity model data (so-called *function-approximation surrogates*, (Queipo et al., 2005; Simpson et al., 2001; Smola and Schölkopf, 2004) or by employing a physics-based *low-fidelity* model (Søndergaard, 2003), a less accurate but computationally much cheaper representation of the original high-fidelity model while being based upon the same physics. The latter approach is used in this paper.

Since the accuracy of the "pure" low-fidelity models is usually not sufficient to directly replace the original high-fidelity model in an optimization loop, it is often necessary to use suitable correction techniques to reduce the misalignment between the low- and high-fidelity model outputs. More specifically, at each iteration of the SBO algorithm, the surrogate is built from the low-fidelity model by aligning its output with the one from the high-fidelity model. A subsequent estimate of the optimal parameter set, a prediction of the high-fidelity model optimum, is then obtained by optimizing this surrogate. A further iteration is required when the low- and high-fidelity model differ "too much" and a new correction operator has to be built. This process of updating the surrogate and its subsequent optimization is iterated in order to keep the surrogate close to the original high-fidelity model. This ensures that the result of the optimization is close to that of the high-fidelity model optimum, while most model evaluations require only the computationally less expensive surrogate model. The proposed SBO algorithm in this paper requires only *one* of the typically expensive high-fidelity model evaluations to construct the surrogate at each iteration. The optimization of the surrogate within each iteration employs evaluations of the computationally cheaper low-fidelity model only. As a result, the presented SBO approach can be very efficient in terms of total number of function evaluations of the high-fidelity model necessary to yield a reasonably accurate optimum, when compared to a far more expensive or even unfeasible direct optimization of the high-fidelity model.

One SBO approach has already been successfully applied on parameter optimization in a biogeochemical model of NPZD type in a single water column (cf. Prieß et al., 2011,2013a,b). The reduction in computation time achieved there was up to 95% compared to conventional parameter optimization. Motivated by these encouraging results, we now extend our initial investigations about SBO. In this paper, the SBO method – for the first time – is applied to a three-dimensional biogeochemical model. Since the source-minus-sink terms in our example model and in most other biogeochemical models are non-differentiable, standard convergence results for the optimization (Conn et al., 2000; Koziel et al., 2010) can not be directly applied. Thus, from the viewpoint of mathematical optimization, it is essential to validate the SBO method for the three-dimensional application numerically. Moreover, when using real measurement data, one would generally have to deal with uncertainties regarding (a) the model's structural complexity required to reconstruct given real measurement data, (b) errors in the measurements and (c) the performance of the optimization method itself. To initially test a necessary precondition for the suitability of the proposed SBO approach, we therefore disregard these uncertainties and conduct twin-experiments with model generated twin data as a first step. Since in these test runs, the solution is known, we are able to assess the feasibility of the method as well as to check the appropriateness and potential for further tuning of inherent parameters of the SBO algorithm. In this regard, it is also reasonable to apply the SBO method initially to a global biogeochemical model of low structural complexity. Feasibility of the SBO approach is proven by demonstrating whether SBO is able to yield a solution close to one of the high-fidelity model optimization (i.e., for the attainable twin data considered here, close to the target data). These experiments yield essential information on the functionality of the proposed approach before investigating other, yet more complex, models and real observational data in a second step.

More specifically, we exploit a simple, albeit still used (cf. Kriest et al., 2012; Parekh et al., 2006) three-dimensional global biogeochemical model simulating the transport and biogeochemical cycles among the single nutrient (N), phosphate, and dissolved organic phosphorus (DOP). A classical spin-up is performed running the model into a steady annual cycle. In this paper, a physics-based low-fidelity model is constructed in the form of a truncated spin-up while considering a fully converged solution for the high-fidelity one. This approach is simple to implement for any model. Moreover, it has the added benefit – compared to for example using a coarser spatial or temporal discretization – that both the high- and the low-fidelity model solution are given on the same temporal and spatial grid, which makes the technical formulation and implementation of the SBO algorithm quite straightforward. We furthermore use a special multiplicative alignment or correction technique to extrapolate the rather inaccurate low-fidelity model onto the original one. This corrected model is the surrogate which we use in a SBO algorithm.

Overall, the results of this first case are very promising and suggest the applicability of the specific SBO approach to real data and also more complex marine ecosystem models as well as the investigation of yet other potential approaches in the context of SBO.

The structure of the paper is as follows: In Section 2, we briefly describe the general structure of marine ecosystem models, the N-DOP model considered as an example in this paper and the numerical solution method based on the TMM. The cost function and the "synthetic" measurement data that we consider for the optimization are described in Sections 3 and 4. In Section 5 we briefly explain the conventional optimization approach that is typically used. Section 6 then descibes the SBO approach, our choice of the low-fidelity model and the corresponding correction approach. The details of the SBO setup are described in Section 7. Numerical results of a SBO run and, for comparison, of a pure low-fidelity model optimization are provided in Section 8. Section 9 covers a discussion of the proposed approach and an outlook to extensions to also more complex models and real measurement data. All further information helpful to understand intermediate steps necessary to obtain the proposed SBO approach can be found in Appendices A, B, C, D.

2. Model description

Coupled biogeochemistry-circulation models consist of a component describing the ocean circulation and a component that describes the biogeochemical source and sink terms. The latter require some representation of the action of marine biology, where ecosystem dynamics are considered. Ocean biota plays an import role in processing, and sometimes effectuating transport of, various climatically relevant chemical elements. A detailed mechanistic understanding of the underlying biological processes is often not available, so that these processes require extensive parameterization in empirical models which, in turn generally depend on a number of difficult to determine parameters. In this respect, parameter identification and optimization can considerably help optimizing, assessing and potentially improving marine biogeochemical models.

2.1. Coupled marine biogeochemical-circulation models

A *fully coupled* marine biogeochemical-circulation model is a system of equations for modeling the ocean circulation including temperature and salinity distributions coupled to equations governing transport and reaction of biogeochemical tracers. The coupling reflects the fact that tracer concentrations are advected by the ocean circulation, their diffusion is dominated by the turbulent mixing of water and, vice versa, a tracer concentration may affect the ocean circulation. A fully coupled simulation is computationally expensive since the simulation of both systems must be performed simultaneously. A single model evaluation in three space dimensions can be performed on high-performance computers only, even more if steady annual cycles – whose simulation requires long-term spin-ups – are looked for.

In contrast, an *off-line* model – as used in this paper – is a simplified approach for tracers that are (or are regarded as) passive, i.e., they do not affect the ocean physics or this influence is neglected. This results in a one-way coupling only, namely from the ocean circulation to the tracer dynamics, i.e., pre-computed circulation fields enter the tracer transport equations as forcing terms. These are the advection velocity vector field $\mathbf{v} = \mathbf{v}(\mathbf{x}, t)$, mixing coefficient $\kappa = \kappa(\mathbf{x}, t)$, temperature and optionally salinity. Here $(\mathbf{x}, t)$ denotes a point in the space–time cylinder $\Omega \times [0, T]$ with $\Omega \in \mathbb{R}^3$ being the spatial domain (i.e., the ocean) with boundary $\Gamma = \partial\Omega$ and $[0, T], T > 0$, the time interval.

With this data given, the marine biogeochemical model considered in an offline computation is the following system of parabolic partial differential equations (here for n tracers, with $y_i = y_i(\mathbf{x}, t)$):

$$\frac{\partial y_i}{\partial t} = A(\kappa, \mathbf{v})\, y_i + q_i(y_1, \ldots, y_n, \mathbf{u}), \quad i = 1, \ldots, n, \tag{1}$$

where $A(\kappa, \mathbf{v})$ is the (time-dependent) linear operator comprising the whole transport, i.e., diffusion and advection, for the given ocean circulation data κ and $\mathbf{v}$. Here, we neglect the additional dependency on the space and time coordinates in the notation for brevity. Note that A is identical for all tracers if the molecular diffusion of the tracers is small compared to the turbulent mixing, which is a reasonable simplification.

Additionally, homogeneous Neumann boundary conditions on Γ for all tracers y_i are imposed. The source-minus-sink terms q_i in general are nonlinear. Usually each of the q_i depends on several other tracers, reflecting the coupling between them. The q_i also include the model parameters (such as growth and mortality rates, sinking velocities etc.) that are subject to identification. They are spatially and temporally constant and summarized in the vector $\mathbf{u} \in \mathbb{R}^m$.

2.2. Transport matrices

We use the so-called *Transport Matrix Method (TMM)* which was introduced by Khatiwala et al. (2005) to compute the effect of the ocean circulation on tracer distributions. Instead of using ocean circulation data $\kappa, \mathbf{v}$ itself and discretizing the corresponding diffusion and advection operators in the tracer transport simulation, the TMM builds up a certain number of temporally averaged matrices A using an ocean model with its numerical diffusion and advection scheme. A comprehensive discussion of the temporal and spatial discretization as well as the process of evaluating transport matrices, especially in combination with operator splitting schemes can be found in Khatiwala et al. (2005).

For the discretized version of (1) we denote by $\mathbf{y}_j$ the appropriately arranged vector of the values $(y_i(\mathbf{x}_k, t_j))_{i,k}$ of all n tracers on all spatial grid points $\mathbf{x}_k \in \Omega$ at the time step j. In the same way, we denote the vector of the discretized source-minus-sink terms q_i at all spatial grid points $\mathbf{x}_k$, evaluated at fixed time t_j, by $\mathbf{q}_j(\mathbf{y}_j, \mathbf{u})$. Using the TMM including a fixed time step τ the time integration scheme reads

$$\mathbf{y}_{j+1} = \mathbf{A}_{imp,j}\,(\mathbf{A}_{exp,j}\,\mathbf{y}_j + \tau\,\mathbf{q}_j(\mathbf{y}_j, \mathbf{u})) =: \varphi_j(\mathbf{y}_j, \mathbf{u}), \quad j = 0, \ldots, n_\tau - 1. \tag{2}$$

Here n_τ is the total number of time steps and $\mathbf{A}_{imp,j}, \mathbf{A}_{exp,j}$ are the implicit and explicit transport matrices at time step j. The matrices are block-diagonal and usually sparse, depending on the used numerical scheme of the ocean model. Starting from a vector $\mathbf{y}_0$ of initial values for the tracers, each step in the time integration scheme just consists of the evaluation of the source-minus-sink term and two matrix–vector multiplications.

For our results we used twelve implicit and twelve explicit transport matrices, which represent monthly averaged tracer transport. The matrices are interpolated linearly to the corresponding discrete time step during simulation (Khatiwala et al., 2005). The latitudinal and longitudinal resolution of the underlying ocean model grid is 2.8125°, including 15 vertical levels. In the following we set the number of steps per year to $n_\tau = 45$. Assuming 360 days a year this time step corresponds to 192 h. Both, the time step and the step count is kept fixed for our analysis and hence is not explicitly specified again.

2.3. The N-DOP model as a biogeochemical model example

As an example application using the SBO method we focus on a three-dimensional coupled marine biogeochemical-circulation model simulating the transport and biogeochemical cycles among the single nutrient (N), phosphate, and dissolved organic phosphorus (DOP), where particulate organic phosphorus, its vertical transport and remineralization are treated implicitly. In the following, we will refer to this as N-DOP model (cf. Kriest et al., 2010). The choice of this specific model has been motivated by the fact that it is a rather simple representative of the class of global marine biogeochemical models. It has been found to yield very similar representations of global biogeochemical tracer distributions as the more complex models (cf. Kriest et al., 2010) and has thus been exploited for diverse model experiments (cf. Kriest et al., 2012; Kwon et al., 2009; Parekh et al., 2006). The considered model therefore provides a suitable first application to prove feasibility of the proposed optimization methodology.

In the following we provide a short model description and follow the notation of Kriest et al. (2010), where more details can be found. The two tracers incorporated in the model will be denoted by y_N and y_{DOP}, respectively. Biological production (in the context of N and DOP biological production is close to net community production) is calculated as a function f of nutrients and light I. The production is limited using a half saturation function, also known as Michaelis–Menten kinetics, and a maximum production rate parameter α as

$$f(y_N, I) = \alpha \frac{y_N}{y_N + K_N} \frac{I}{I + K_I}.$$

Light, here, is a portion of short wave radiation I_{SWR}, which is computed as a function of latitude and season following the astronomical formula of Paltridge and Platt (1976). The portion depends on the photo-synthetically available radiation σ_{PAR}, the ice cover σ_{ice} and the exponential attenuation of water

$$I = I_{SWR}\,\sigma_{PAR}\,(1 - \sigma_{ice})\,\exp(-z K_{H2O}),$$

where z denotes the vertical coordinate and K_{H2O} is the attenuation coefficient for sea water. A fraction of the biological production σ remains suspended in the water column as dissolved organic phosphorus, which remineralizes with a rate λ. The remainder of the production sinks instantaneously as particulate matter to depth where it is remineralized instantaneously according to the empirical power law relationship determined by Martin et al. (1987). Similar approaches of modeling biological production can be found in Dutkiewicz et al. (2005), Parekh et al. (2005) and Yamanaka and Tajika (1997).

Table 1
Element in parameter vector, variable name, description and units for the N-DOP model parameters.

u_i	Name	Description	Unit
u_1	λ	remineralization rate of DOP	1/d
u_2	α	maximum community production rate	1/d
u_3	σ	fraction of DOP	–
u_4	K_N	half saturation constant of N	mmol P/m^3
u_5	K_I	half saturation constant of light	W/m^2
u_6	K_{H2O}	attenuation of water	1/m
u_7	b	sinking velocity exponent	–

The water column is divided into a productive euphotic zone Ω_1 with a depth z' of about 100 m, and an aphotic zone Ω_2 below. The biological source-minus-sink terms can then be written as

$$q_N(y_N, y_{DOP}, \mathbf{u}) = \begin{cases} -f(y_N, I) + \lambda y_{DOP} & \text{in } \Omega_1 \\ (1-\sigma)\,\frac{\partial}{\partial z} F(y_N, I) + \lambda y_{DOP} & \text{in } \Omega_2 \end{cases}$$

$$q_{DOP}(y_N, y_{DOP}, \mathbf{u}) = \begin{cases} \sigma f(y_N, I) - \lambda y_{DOP} & \text{in } \Omega_1 \\ -\lambda y_{DOP} & \text{in } \Omega_2 \end{cases}$$

where

$$F(y_N, I) = (z/z')^{-b} \int_0^{z'} f(y_N, I)\, dz.$$

The model parameters to be identified are summarized in the vector $\mathbf{u}$ and given in Table 1.

2.4. Computation of a steady annual cycle

In our exemple, we use precomputed ideal or synthetic data denoted by $\mathbf{y}_d$ that have been generated by running the model into a steady annual cycle. The spin-up consists of a repeated application of the mapping Φ, where $\Phi := \varphi_{n_\tau - 1} \circ \ldots \circ \varphi_0$ with the φ_j defined in (2). In this setting one application of Φ corresponds to the computation of one year model time. We set

$$\mathbf{y}^{l+1} = \Phi(\mathbf{y}^l, \mathbf{u}), \quad l = 0, \ldots, n_l - 1, \tag{3}$$

where n_l is the total number of iterations (model years) necessary to compute a steady annual cycle and $\mathbf{y}^l$ denotes the vector of discretized tracer after l years. The iteration starts with a constant distribution $\mathbf{y}^0$ of both tracers, 2.17 mmol P/m^3 and 10^{-4} mmol P/m^3.

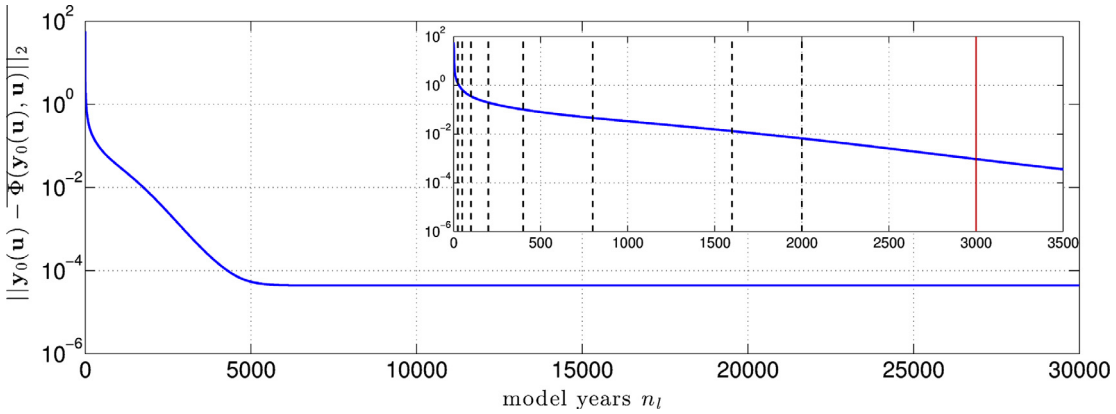


Fig. 1. Convergence of the spin-up towards a solution $\mathbf{y}(\mathbf{u})$ for some illustrative parameter vector $\mathbf{u}$. Shown is the Euclidean norm of the residual (cf. Section 2.4) using a semi-log scale. Inset: detailed section. In this paper, we consider a reduced number of model years n_l to create a low-fidelity model. Investigated low-fidelity models (cf. Appendix B.1) are indicated by vertical dashed black lines (the one with $n_l = 25$ is finally used for SBO). The solution after $n_l = 3000$ model years (vertical red line) is considered as the reference high-fidelity model solution. (For interpretation of the references to colour in this figure caption, the reader is referred to the web version of this article.)

The model integration is implemented as part of the simulation package of Metos3D (Marine Ecosystem Toolkit for Simulation and Optimization in 3-D), see (Piwonski and Slawig, 2011). From several computations it can be observed that after $n_l = 3000$ iterations (model years), a numerical steady solution (up to an accuracy of more than 10^{-2} in Euclidean norm, compare Fig. 1) is obtained. The residual can be further decreased by using a higher number n_l of model years used in the spin-up (3). However, the number $n_l = 3000$ of steps (already used for example in Kriest et al. (2010)) provides a satisfactory accuracy. Thus we refer to this as a converged steady annual cycle and take it as the reference high-fidelity model solution. In the following we add the subscript f to distinguish the high-fidelity model state and corresponding number of model years, i.e., $\mathbf{y}_f, n_{f,l}$, from the corresponding low-fidelity model ones.

3. Cost function

For the optimization, we define the following discrete cost function by measuring the difference between the actual model solution and the target in a squared Euclidean norm as

$$J(\mathbf{y}(\mathbf{u})) := \frac{1}{2}\,\|\mathbf{y}(\mathbf{u}) - \mathbf{y}_d\|_2^2 = \sum_{i=1}^{2}\sum_{j=1}^{p}\left(y_{i,j} - (y_d)_{i,j}\right)^2, \tag{4}$$

with the steady state solution $\mathbf{y}$ (cf. Section 2.4) of dimension p and for given parameters $\mathbf{u}$ and where $y_{i,j}$ denotes the value (i.e., the concentration) of tracer i (phosphate and dissolved organic phosphorus) at one discrete point in space and time over one annual cycle. Optional weights may be applied, e.g. proportional to the inverse values of the error variances of the measurements. Because only one datatype (phosphate) is incorporated in the N-DOP model, we do not consider any normalization of the distinct components of the state vector $\mathbf{y}$. An investigation of different cost function formulations for the N-DOP model can be found in Kriest et al. (2010).

4. Synthetic data

From the viewpoint of mathematical optimization and to reduce the number of uncertainties when dealing with real measurement data, it is essential to initially prove feasibility of the proposed optimization approach by considering an "identical-twin experiment" mode in a first step: Considering the reference high-fidelity model, a ground-truth *attainable* model solution $\mathbf{y}_f$ is chosen for some randomly selected parameter vector, in the following denoted by $\mathbf{u}_d$. From this "true" solution, data $\mathbf{y}_d$ are obtained as

$$\mathbf{y}_d := \mathbf{y}_f(\mathbf{u}_d). \tag{5}$$

Since we use synthetic (i.e., "perfect") data in this paper, optional weights have not been applied to the cost function in (4). Here we assume that the entire state vector $\mathbf{y}_d$ is observed at all time steps during an annual cycle.

A necessary condition for the applicability of the parameter optimization algorithm is that a solution at least close to the target parameter $\mathbf{u}_d$ vector can be obtained given the "synthetic data" $\mathbf{y}_d$. Most importantly, for the SBO to provide a promising calibration tool, it needs to be computationally more efficient when compared to a direct optimization using conventional algorithms.

5. Direct optimization process

In order to identify the parameters in the biogeochemical model, one would typically "plug" the high-fidelity model into the following nonlinear optimization problem (cf. Fig. 2(a))

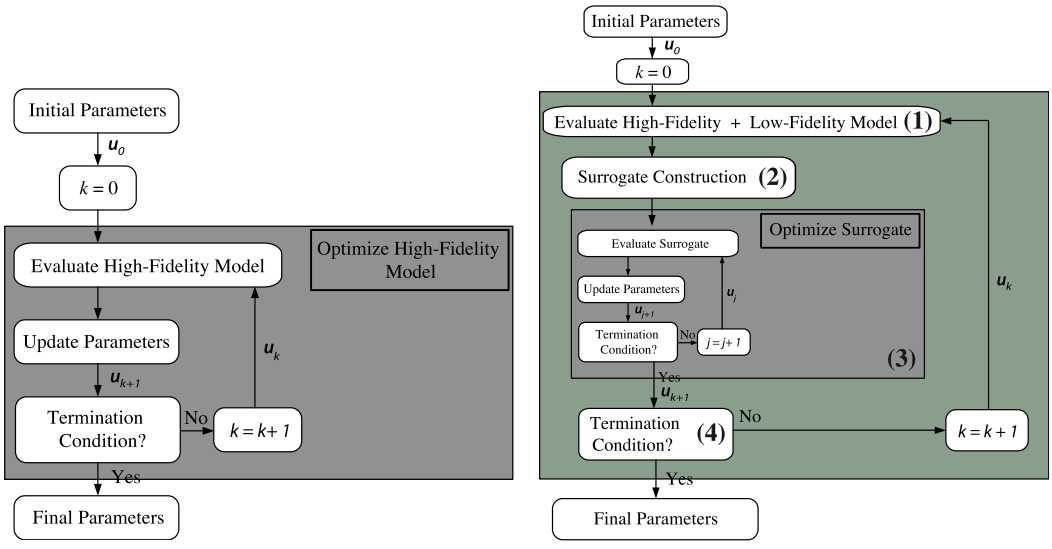
$$\min_{\mathbf{u}\in U_{ad}} J(\mathbf{y}_f(\mathbf{u})), \tag{6}$$

with

$$U_{ad} := \{\mathbf{u} \in \mathbb{R}^m : \mathbf{b}_l \leqslant \mathbf{u} \leqslant \mathbf{b}_u\},$$
$$\mathbf{b}_l, \mathbf{b}_u \in \mathbb{R}^m, \quad \mathbf{b}_l < \mathbf{b}_u,$$

which will be denoted as direct optimization of the high-fidelity model in the following. Here, m denotes the total number of model parameters to be identified (for the N-DOP model we have $m = 7$). The inequalities in the definition of the set U_{ad} of admissible parameters are meant component-wise.

However, when optimizing the expensive high-fidelity model using any conventional approaches (deterministic, e.g., gradient-based or stochastic, e.g., meta-heuristics), typically, the



(a) Direct Optimization, $\mathbf{u}_{k+1} = \underset{\mathbf{u}\in U_{ad}}{\operatorname{argmin}}\, J(\mathbf{y}_f(\mathbf{u}))$.

(b) Surrogate-Based Optimization, $\mathbf{u}_{k+1} = \underset{\mathbf{u}\in U_{ad}}{\operatorname{argmin}}\, J(\mathbf{s}_k(\mathbf{u}))$.

Fig. 2. In a direct optimization (a), the original expensive high-fidelity model $\mathbf{y}_f$ under consideration is directly employed in an optimization loop using conventional optimization approaches. SBO replaces the direct optimization by iterative updating (1 + 2) and re-optimization (3) of its surrogate $\mathbf{s}_k$, a computationally cheaper but yet reasonably accurate representation. Here, $\mathbf{u}_k$ denotes the parameter vector at the outer iteration k.

computational burden is prohibitively large. Thus, we omitt such a direct optimization here.

6. Surrogate-based optimization

SBO replaces the direct optimization (6) of the original expensive high-fidelity model $\mathbf{y}_f$ by iterative updating (steps (1)+(2) in Fig. 2(b)) and re-optimization (step (3) in Fig. 2(b)) of its surrogate, a computationally cheaper but yet reasonably accurate representation. For the optimization of each surrogate, a gradient-based algotihm is used (cf. Section 7). In this paper, the surrogate is constructed from a physics-based so-called *low-fidelity model* (step (1) in Fig. 2(b)). The accuracy of the underlying low-fidelity model itself is not sufficient and its direct optimization therefore does not yield satisfactory results (cf. Section 8). Thus and characteristic for the SBO approach, the low-fidelity model is aligned using a suitable correction technique to reduce its misalignment with respect to the high-fidelity model solution and to ensure that the corrected low-fidelity model (the surrogate) provides a reliable prediction of the high-fidelity model optimum. This correction involves evaluation of the high-fidelity model (step (1) in Fig. 2(b)). The specifics are described below.

6.1. Low-fidelity model

In this paper, we follow the approach of a truncated spin-up. More specifically, we employ a reduced number of model years n_l to solve for an approximation of the steady annual cycle (cf. Section 2.4). The time step τ employed in the underlying time integration scheme (2) is fixed; the same as for the reference high-fidelity model. For the sake of further analysis, the low-fidelity model state and number of model years employed will be denoted by $\mathbf{y}_c$ and $n_{c,l}$, respectively, to be distinguishable from those used for the high-fidelity model, $\mathbf{y}_f$ and $n_{f,l}$. The time required for one low-fidelity model evaluation, compared to the one required for the high-fidelity model, is thus reduced by the factor α_{eval} which is simply given as

$$\alpha_{eval} = (n_{f,l}/n_{c,l}). \tag{7}$$

Fig. 1 shows the convergence of the model integration towards a steady-state annual cycle $\mathbf{y}(\mathbf{u})$ (cf. Section 2.4) as well as the difference between the high- and low-fidelity models models employing distinct number of model years.

6.2. Low-fidelity model correction

We exploit a *multiplicative response correction* to map the low-fidelity model solution onto the high-fidelity one. This approach is motivated by the fact that the overall "shape" of the low-fidelity model solution resembles that of the high-fidelity one, e.g., seasonal maxima and minima occur at similar times, which is the consequence of the low-fidelity model being physics-based. For a more detailed motivation of this approach see Appendix B.2. This technique has already been investigated and successfully applied to a one-dimensional marine ecosystem model in Prieß et al. (2011).

For each iteration k of the optimization process, a multiplicative correction vector $\mathbf{a}_k$ is given as the point-wise (i.e., for each component of the state space) ratio of the high-fidelity to the low-fidelity model solution. Steps (1)-(2) in Fig. 2(b) to construct the surrogate $\mathbf{s}_k$ at iteration k of the SBO are then in detail given by

$$\begin{aligned}
&\mathbf{y}_f(\mathbf{u}_k) && \text{Evaluate high-fidelity model}\\
&\mathbf{y}_c(\mathbf{u}_k) && \text{Evaluate low-fidelity model}\\
&\mathbf{a}_k := \frac{\mathbf{y}_f(\mathbf{u}_k)}{\mathbf{y}_c(\mathbf{u}_k)} && \text{Compute correction vector}\\
&\mathbf{s}_k(\mathbf{u}) := \mathbf{a}_k\,\mathbf{y}_c(\mathbf{u}) && \text{Surrogate model construction}
\end{aligned} \tag{8}$$

Table 2
Initial, optimal and final parameters $\mathbf{u}_0, \mathbf{u}_d$ and $\mathbf{u}_s^*$ for the SBO run (cf. Fig. 2(b)). Optimized parameters $\mathbf{u}_k$ are determined by the stopping criterion (9). The iterates $\mathbf{u}_2, \mathbf{u}_5$ and $\mathbf{u}_{10}$ correspond to thresholds $\gamma = 5\cdot10^{-2}, 5\cdot10^{-3}, 5\cdot10^{-4}$. For units see Table 1.

iterate	u_1	u_2	...				u_7
$\mathbf{u}_0$	0.3	5.0	0.4	0.8	25	0.04	0.78
$\mathbf{u}_2$	0.502	3.328	0.633	0.845	24.886	0.036	0.92
$\mathbf{u}_5$	0.482	2.562	0.652	0.856	24.99	0.027	0.885
$\mathbf{u}_{10}$	0.485	2.334	0.659	0.745	25.076	0.025	0.864
$\mathbf{u}_d$	0.5	2.0	0.67	0.5	30.0	0.02	0.858
$\mathbf{b}_l$	0.25	1.5	0.05	0.25	10.0	0.01	0.7
$\mathbf{b}_u$	0.75	200.0	0.95	1.5	50.0	0.05	1.5

where the multiplication to construct the surrogate is again meant point-wise. For the above choice of the model correction, only one evaluation of the expensive high-fidelity model is required to construct the surrogate once at the beginning of each outer iteration k. Each subsequent surrogate optimization involves evaluations of the cheaper low-fidelity model only (plus applying the fixed correction step at zero cost). Since the number of outer iterations is much lower when compared to conventional approaches, the optimization costs are comparably low.

6.3. Further enhancements

For the above surrogate it could occur that the low-fidelity model solution is, for individual components of the state vector, close to zero (or a few magnitudes smaller than the solution of the high-fidelity model). Following the above approach can then lead to large entries in the corresponding correction vector $\mathbf{a}_k$). Resulting "spikes" appearing in the surrogate's solution may be viewed as numerical noise that slows down the algorithm's convergence and makes the high-fidelity model optimum more difficult to locate.

We apply some simple modifications that allow us to eliminate potentially adverse impacts of the problems described above. These modifications do not require any extra computational overhead, and include: (i) upper and lower bounds $a_{\text{ub}}, a_{\text{lb}}$ for the correction factors in $\mathbf{a}_k$, (ii) setting the high- and low-fidelity model solution values to zero (and the correction factor to one) if their values lie below a certain threshold δ, which is supposed to be of the order of the discretization error of the model. For the considered problem, we use $\delta = 5\cdot10^{-3}$ mmol P/ m^3. These simple modifications can further improve the accuracy of the surrogate as well as the performance of the optimization algorithm, which has been investigated in Prieß et al. (2013b) for a similar response correction approach and another marine ecosystem model. In the following we choose the bounds $a_{\text{lb}} = 0.1$ and $a_{\text{ub}} = 5$ which, from numerical experiments, turned out to be a reasonable choice.

7. Optimization setup

The SBO starts from a random parameter vector $\mathbf{u}_0$. Each updated surrogate is optimized (step (3) in Fig. 2(b)) using the MATLAB[1] function `fmincon`, exploiting the active-set algorithm (using the options "'TolCon', 1e-6, 'TolX', 1e-6, 'TolFun', 1e-6"). The specific initial and optimal parameter vector $\mathbf{u}_0$ and $\mathbf{u}_d$, the lower and upper parameter bounds $\mathbf{b}_l$ and $\mathbf{b}_u$ are explicitly given in Table 2. To ensure convergence of the SBO, we enhance each surrogate optimization (step (3) in Fig. 2(b)) employing trust-region convergence safeguards, i.e. by restricting the current step-size to a certain trusted

[1] MATLAB is a registered trademark of The MathWorks, Inc., http://www.mathworks.com.

region δ_k (cf. Appendix A and D). The performance of the SBO process is assessed through investigating the accuracy of matching the target data and optimal parameters (5) by the solutions obtained at each iteration as well as the computational cost. The latter is measured in terms of equivalent high-fidelity model evaluations (cf. Section 7.2).

7.1. Stopping criteria

The process sketched in Fig. 2(b) of aligning the low-fidelity model to obtain the surrogate, subsequent optimization of this surrogate is repeated until a user-defined termination condition is satisfied (step (4) in Fig. 2(b)). For the considered problem, we use the absolute step size (measured in the Euclidean norm) between two successive iterates $\mathbf{u}_k$ and $\mathbf{u}_{k-1}$ as well as a lower bound $\delta_k^{\min}$ for the trust-region radius δ_k. The overall solution $\mathbf{u}_s^*$ of the SBO is thus given as

$$\mathbf{u}_s^* = \mathbf{u}_{k_{min}} \quad \text{with}$$
$$k_{min} := \min\left\{k = 1,2,\ldots : \|\mathbf{u}_k - \mathbf{u}_{k-1}\|_2^2 \leqslant \gamma \ \vee \ \delta_k \leqslant \delta_k^{\min}\right\}. \tag{9}$$

Here, the parameters are non-dimensionalized. Thus, units of $\delta_k^{\min}$ and δ_k are given according to the units in Table 1. It might not be necessary to run the overall SBO until convergence because an approximate solution could already be sufficient as the surrogate model is not perfectly accurate anyway. Thus, using a rather relaxed stopping criterion could allow us to obtain a sufficiently accurate solution at rather low computational cost. To assess a reasonable trade-off between the quality of the SBO solution and the computational costs, we look at distinct iterations in the SBO process using the three thresholds

$$\left\{\gamma, \delta_k^{\min}\right\} = \left\{5\cdot 10^{-2}, 5\cdot 10^{-3}\right\}, \left\{5\cdot 10^{-3}, 5\cdot 10^{-4}\right\}, \ldots \left\{5\cdot 10^{-4}, 5\cdot 10^{-5}\right\}. \tag{10}$$

The optimization of the each surrogate (step (3) in Fig. 2(b)) is terminated after a specific number of iterations, here we use 10 iterations. The reason is the same as for limiting the number of outer SBO iterations. We should also point out that choosing the above termination conditions are up to the user and is generally problem dependent.

7.2. Optimization cost

This cost of the SBO is measured in terms of the total number of equivalent high-fidelity model evaluations. Compared to the cost of a high-fidelity model evaluation, the cost of evaluating the low-fidelity model is reduced by the factor α_{eval} as given in (7). The cost of one iteration of the SBO procedure (one k loop in Fig. 2(b)) equals the number of low-fidelity model evaluations necessary to optimize the surrogate model divided by this factor α_{eval}, and increased by one since only one high-fidelity model evaluation is required for the correction of the low-fidelity model solution (step (1) in Fig. 2(b)).

7.3. Low-fidelity model optimization

To explore the benefit of applying the correction step (cf. Section 6.2) in the surrogate approach, we also perform a direct optimization of the "pure" low-fidelity model, i.e. without applying any correction to improve its accuracy. In a first experiment, we employ the given number of model years for the selected low-fidelity model for each step within the optimization. Here, as for the SBO, the iteration always starts with a constant distribution of all tracers (cf. Section 2.4). In a second experiment, we again perform the optimization of the low-fidelity model, while each spin-up is now started from the previously calculated model solution.

8. Results

8.1. Initial investigation

An initial investigation established confidence that the low-fidelity model using $n_{c,y} = 25$ spin-up iterations is a reasonable choice to construct a reliable surrogate in conjunction with the multiplicative response correction approach as described in Section 6.2 (see Appendix B.2 and C for details). This is illustrated in Fig. 3. Obviously, the chosen multiplicative correction technique significantly improves the accuracy of the low-fidelity model. Shown are the low-, the high-fidelity model and the surrogate's simulated phosphate concentrations at two selected parameter vectors $\bar{\mathbf{u}}_k, \tilde{\mathbf{u}}_k$ in the neighborhood of one point $\mathbf{u}_k$ where the correction has been established (cf. (8)). Parameter values $\mathbf{u}_k$ are taken from one iteration of an illustrative SBO run. The smaller the neighborhood, the closer the surrogate lies to the desired high-fidelity model output (cf. Appendix C). This agreement of the surrogate with the reference high-fidelity model is important for the performance of the SBO, i.e., how fast it converges to a solution (cf. Appendix A). The surrogate model defined in (8) satisfies, by definition, zero-order consistency at the point of alignment $\mathbf{u}_k$, i.e.,

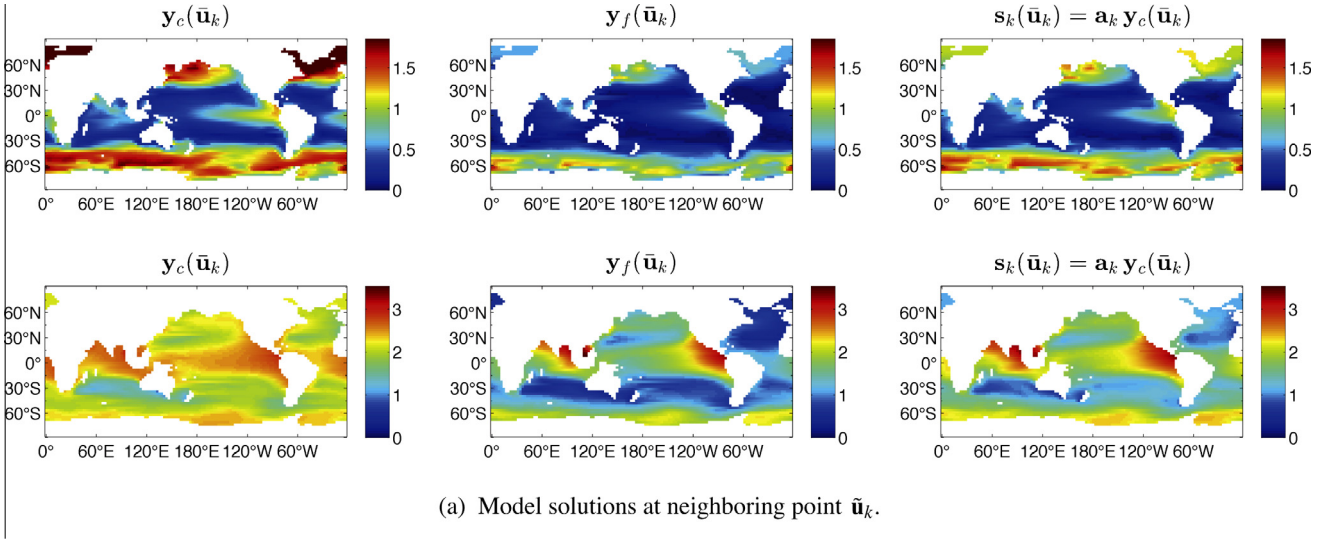
$$\mathbf{s}_k(\mathbf{u}_k) = \mathbf{y}_f(\mathbf{u}_k). \tag{11}$$

Our surrogate model does not use high-fidelity model derivatives. Hence, consistency in model sensitivities around $\mathbf{u}_k$ cannot be satisfied exactly. Nevertheless, as shown in Fig. 3, the surrogate provides a reasonable approximation of the high-fidelity model also in the neighborhood of $\mathbf{u}_k$. This is a result of the surrogate being physics-based which leads to derivatives that are expected to be at least similar to those of the high-fidelity model.

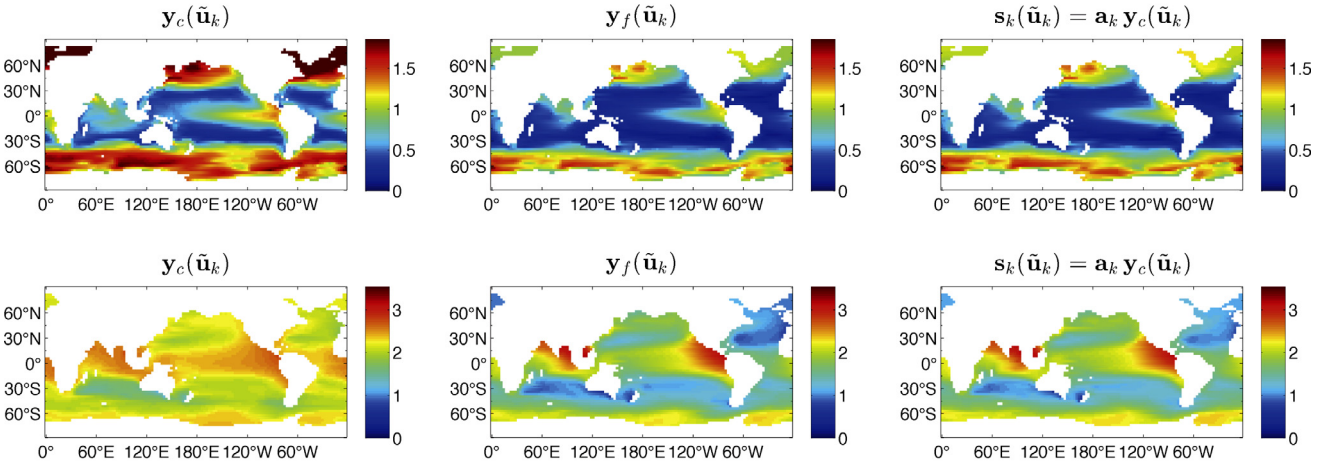
8.2. Optimization

In a second step, the operation and performance of the proposed SBO is illustrated for an "identical-twin" experiment with the reference high-fidelity model using $n_{f,y} = 3000$ spin-up iterations (cf. Section 2.4). The quality of the results is evaluated by assessing the model-data differences in terms of (a) the improvement in the reference cost function $J(\mathbf{y}_f)$ (cf. (4)), i.e., the one employing the reference high-fidelity model, and (b) in terms of matching the target phosphate distribution as well as optimal parameters (cf. (5)). Figs. 4 and 5 show corresponding convergence plots for the reference cost function value and the step size norm (both versus number of iterations and equivalent number of high-fidelity model evaluations) and for the single parameter values $u_{k,i}$. Figs. 6 and 7 present the high-fidelity model solution for iterates $\mathbf{u}_2, \mathbf{u}_5$ and $\mathbf{u}_{10}$ obtained in the SBO (corresponding to different stopping criteria in (9)). Table 2 shows the corresponding parameter values.

After two iterations of the SBO, the solution $\mathbf{y}_f(\mathbf{u}_2)$ (corresponds to a threshold of $\gamma = 5\cdot 10^{-2}$ in (9)) already resembles the main patterns of the target distribution $\mathbf{y}_d$ (cf. Figs. 6 and 7). Two out of seven model parameters can be identified at this iteration (cf. Fig. 5). After five iterations (corresponding to $\gamma = 5\cdot 10^{-3}$), both the agreement of the solution $\mathbf{y}_f(\mathbf{u}_5)$ with the target data and the parameter match are further improved. This improvement is reflected by the further reduction in the cost function value by one order of magnitude (cf. Fig. 4). A threshold of $\gamma = 5\cdot 10^{-4}$ is reached after ten iterations in the SBO process, with a solution $\mathbf{y}_f(\mathbf{u}_{10})$, but yields only slight improvements over the one obtained after five iterations. For the chosen set-up, the method can identify 5 out



(a) Model solutions at neighboring point $\tilde{\mathbf{u}}_k$.



(b) Model solutions at closer point $\tilde{\mathbf{u}}_k$.

Fig. 3. Shown are – from left to right columns – the low-, the high-fidelity model and the surrogate's simulated phosphate concentrations $\mathbf{y}_c, \mathbf{y}_f$ and $\mathbf{s}_k$ at the surface (upper rows in (a) and (b)) and at 455 m depth (lower rows in (a) and (b)) in January. The chosen multiplicative correction technique significantly improves the accuracy of the low-fidelity model. In the point of alignment $\mathbf{u}_k$, the surrogate and high-fidelity model coincide (plots are omitted here).

of 7 parameters with a solution close to the target data (see also the discussion in Section 9). The crucial point is that these solutions of the SBO approach can be obtained at the cost of a very few high-fidelity model evaluations: In our identical-twin example, the equivalent of 8, 19 and 38 high-fidelity model evaluations were required to obtain the solutions $\mathbf{u}_2, \mathbf{u}_5$ and $\mathbf{u}_{10}$.

In contrast to the successfully applied SBO, illustrative optimization runs of the pure low-fidelity model – both with- and without restarting from the state-vector solution of the previous iteration (cf. Section 7) – are computationally even cheaper (since no high-fidelity model data is used at all) but do not yield satisfactory results at all. The quality of the obtained solution is low, which is also reflected by the still high cost function value for the obtained parameters. Without restarting from the state vector of the previous iteration, a cost function value of $J(\mathbf{y}_f) \approx 9.5 \cdot 10^4$ is obtained. When iterations are restarted from the state vector of the previous iteration, a similar solution is obtained, corresponding to a cost function value of $J(\mathbf{y}_f) \approx 4 \cdot 10^5$. We omit to show the solutions obtained by these low-fidelity optimizations in more detail since they are hardly closer to the target than the initial guess $\mathbf{y}_f(\mathbf{u}_0)$. On the other hand, SBO yields a much more accurate solution corresponding to a significantly lower cost function value of $J(\mathbf{y}_f) \approx 2 \cdot 10^2$ (cf. Fig. 4). The poor performance of the pure low-fidelity model optimization runs is likely related to inaccurate gradient information, which can differ substantially from the one of the high-fidelity model. In contrast, the gradient of the corrected low-fidelity model, the surrogate, is generally much closer to that of the high-fidelity model, which explains the much better performance of the SBO runs compared to the optimizations using the low-fidelity model without correction step. For illustration, Table 3 shows the gradient of the low-fidelity, the high-fidelity model and the surrogate at the initial parameter vector $\mathbf{u}_0$.

Overall, the SBO approach yields a computationally very efficient optimization procedure. Because of computational requirements, a direct optimization of the high-fidelity model was not performed, but its cost can be estimated based on typical numbers necessary to optimize the surrogate model. More specifically, within the SBO, the typical number of linesearch steps performed were five to ten. Considering seven evaluations required for each gradient using finite differences (the model includes seven model parameters), 12 to 17 model evaluations were thus in total necessary for each iteration in the optimization. Assuming typical numbers of 30 to 100 iterations for a direct optimization using standard optimization routines the latter would presumably require a few

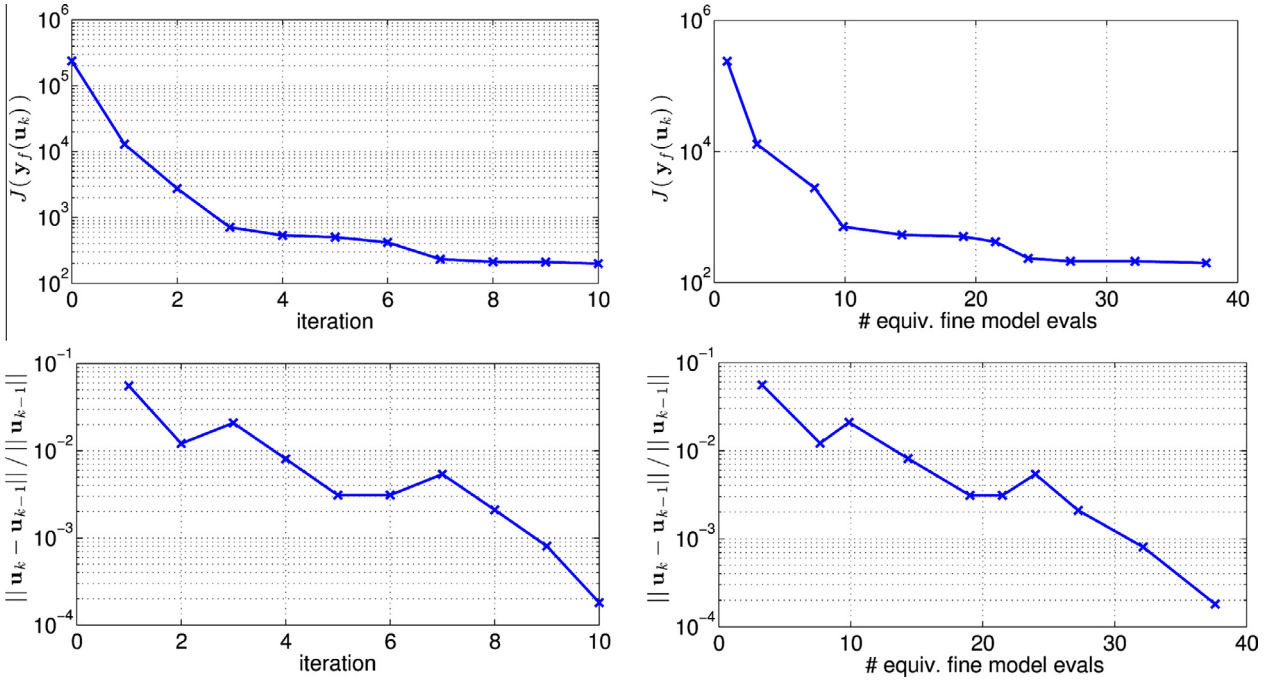


Fig. 4. Convergence of the cost function value $J(\mathbf{y}_f)$ (cf. (4)), the step-size norm and of the trust-region radius δ_k (semi-log scale and both versus number of iterations and equivalent number of high-fidelity model evaluations) for the exemplary SBO run. Updated trust-region radii according to (D.2) and (D.3) are shown.

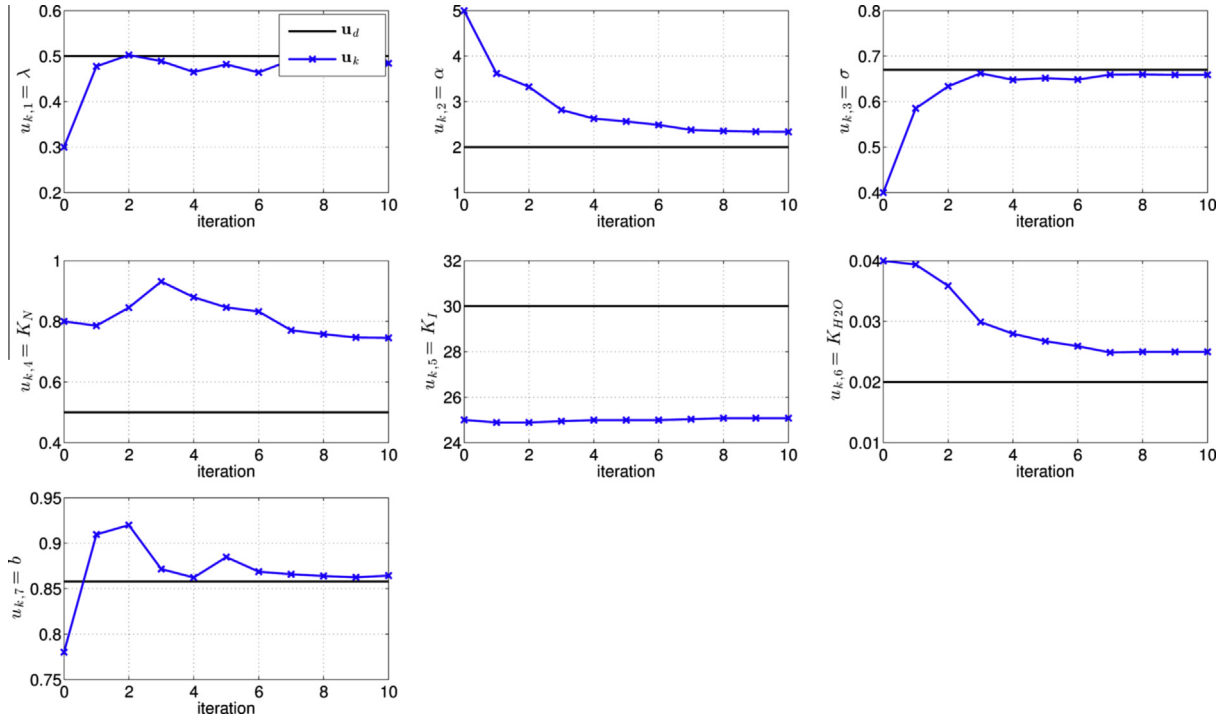


Fig. 5. Convergence of the single parameter values $u_{k,i}$ for each iteration k of the exemplary SBO run.

hundred to thousands of model evaluations (depending on the required accuracy and hence number of iterations performed). Assuming approximately 30 min for a single high-fidelity model evaluation on a 48-processor cluster, a direct optimization approach would then require several days to weeks. On the other hand, the SBO solution requires approximately 4 to 19 h (again, depending on the required accuracy and plus some overhead associated with input–output routines).

9. Discussion and outlook

9.1. Performance of current approach

We here do not use information about the high-fidelity model solution's sensitivity to parameter variations in the construction of the surrogate (cf. (11)). Thus, 1st-order consistency is not exactly ensured (see, e.g., Table 3) and the obtained accuracy in

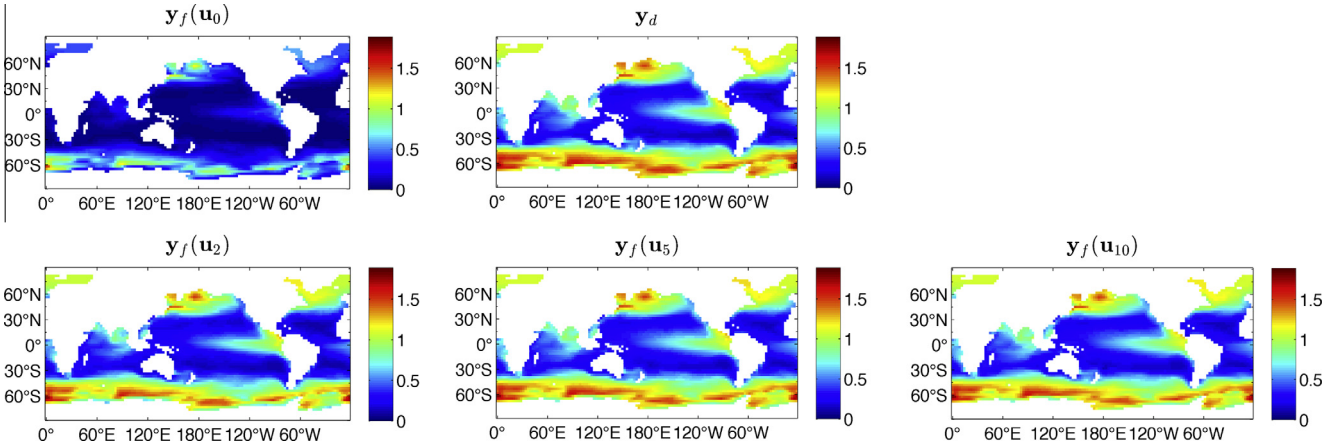


Fig. 6. High-fidelity model output $\mathbf{y}_f$ evaluated at the initial parameter vector $\mathbf{u}_0$ and at the parameters $\mathbf{u}_2, \mathbf{u}_5, \mathbf{u}_{10}$ obtained by the exemplary SBO run after two, five and ten iterations (corresponding to different thresholds employed in the termination condition in (9)) as well as the target data $\mathbf{y}_d$, for surface phosphate in January.

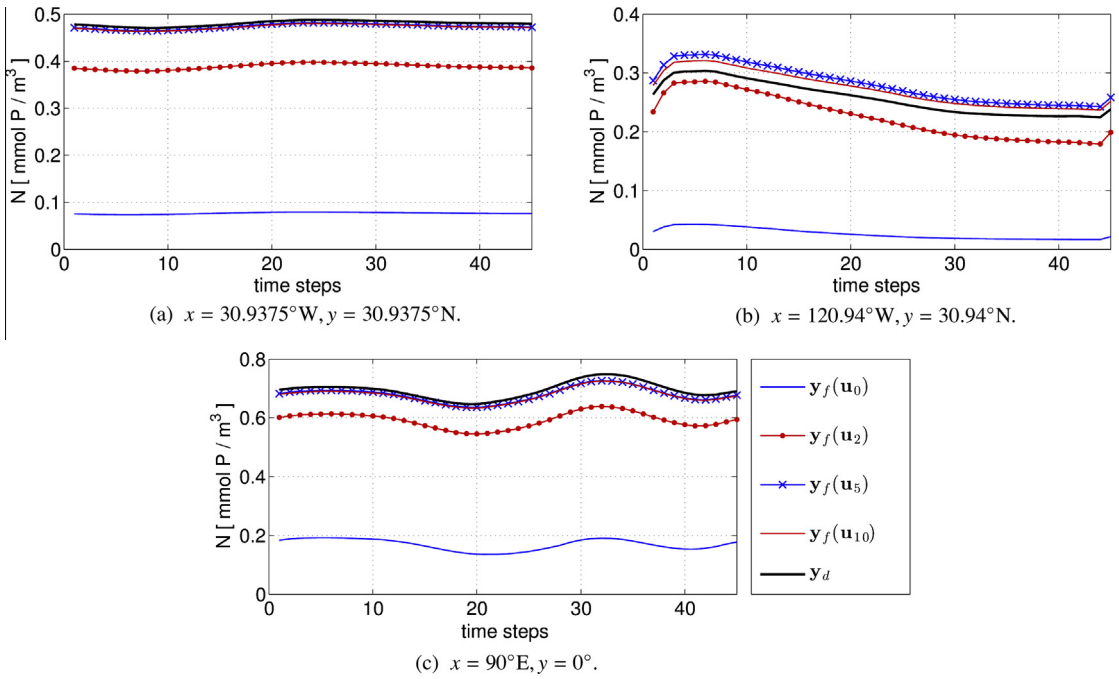


Fig. 7. Corresponding to Fig. 6. Shown here are the annual cycles of surface phosphate at three distinct locations.

Table 3
Gradient of the low-, the high-fidelity model and of the surrogate, here, for illustration, at the initial parameter vector $\mathbf{u}_0$. For most components, the correction of the low-fidelity model implicit in the surrogate approach improves the gradient with respect to that of the low-fidelity model. For units see Table 1.

	u_1	u_2	...				u_7
$\mathbf{y}_c'(\mathbf{u}_0)$	$3.52 \cdot 10^4$	$4.21 \cdot 10^3$	$-3.87 \cdot 10^4$	$-1.83 \cdot 10^4$	$-3.15 \cdot 10^2$	$-1.39 \cdot 10^5$	$1.19 \cdot 10^5$
$\mathbf{y}_f'(\mathbf{u}_0)$	$1.56 \cdot 10^5$	$5.63 \cdot 10^4$	$-8.29 \cdot 10^5$	$-2.02 \cdot 10^5$	$-6.29 \cdot 10^3$	$-6.10 \cdot 10^6$	$-9.24 \cdot 10^5$
$\mathbf{s}_0'(\mathbf{u}_0)$	$-4.87 \cdot 10^4$	$1.95 \cdot 10^4$	$-1.25 \cdot 10^5$	$-5.44 \cdot 10^4$	$-2.21 \cdot 10^3$	$-2.04 \cdot 10^6$	$-1.80 \cdot 10^5$

terms of identifying the correct parameters is not perfect (cf. Table 2 and Fig. 5). This may be related to the low sensitivity of the model with respect to some of the parameters, which has already been discovered in Rückelt et al. (2010) for another marine ecosystem model. A similar sensitivity investigation for the N-DOP model would be useful. Also, the reason for the mismatch in some parameters might be the possibly too low accuracy of the underlying low-fidelity model which we selected here.

The consistency of the surrogates' gradient with the one of the high-fidelity model could be further improved by choosing a more accurate low-fidelity model (i.e., a larger number of model years) or even ensured exactly by including high-fidelity model sensitivity data in the surrogate's construction. On the one hand, this could allow locating the solution even more precisely particularly in terms of model parameters. On the other hand, this would increase the overall optimization costs. Thus, the current solution is to some extent pragmatic and provides a reasonable trade-off between

accuracy of the solution and low optimization costs. A further investigation of these aspects could be beneficial as part of future research.

In conclusion, the SBO approach of combining a multiplicative response correction approach with a low-fidelity model is able to yield a solution at low computational cost when compared to standard optimization routines. The main characteristics of the target data can be reproduced while 5 out of 7 model parameters have been identified.

9.2. Extensions to more complex models and real data

Overall, the results of this first case study demonstrate the principal feasibility of the proposed optimization approach and suggest its applicability to more complex biogeochemical models and real measurement data.

Clearly, when dealing with real data, further uncertainties would have to be taken into account. Measurements might be incomplete and contain errors, and a model solution that is fully consistent with the data may not exist. However, "wrong" model structure and complexity would be a problem of the high-fidelity model iself and of any conventional optimization approach. The surrogate, however, is always constructed with respect to the – current state-of-the-art – high-fidelity model and a well-performing SBO always yields a solution close to a minimum of the high-fidelity model optimization (at low computational costs). The obtained solution would then be the current best fit of the data. Assuming a reference model with a higher cabability to reconstruct the data would be available, SBO concepts developed and insights gained in previous investigations could then be transferred and used to calibrate models and eventually guide and demonstrate model improvement.

Regarding the appropriateness of the considered N-DOP model in conjunction with the TMM approach, our case study presents an initial test of the SBO approach. The use of the TMM – rather than a fully coupled circulation model – could imply a more linear model behaviour. However, it was found by Khatiwala et al. (2005) that the model solution using the TMM is very similar to the one obtained when using a full circulation model.

Furthermore, a structurally more complex biogeochemical model, for example of NPZD type, would clearly include a higher number of non-linearities which could affect the efficacy of the SBO technique. However, first investigations about SBO considering a one-dimensional biogeochemical model of NPZD type have already been completed in Prieß et al. (2011,2013a,b). Therein, the nonlinear model output has been handled by using a smoothing technique, for which SBO was demonstrated to be succesful. Applications of the SBO approach to structurally more complex three-dimensional models will be investigated in a forthcoming study.

When extending our initial approaches to a range of models, alternative ways to construct a physics-based low-fidelity model might be also promising. This could include, for example, a coarsening in the spatial and/or temporal resolution. Also, a combination of the truncated spin-up proposed here with the aforementioned coarsening approaches might be possible and computationally efficient. Obtaining a low-cost optimization with a high-quality solution (both in terms of the optimal model solution and parameter match) is the fundamental aim of SBO. Finding a reasonable trade-off between these two objectives will be of central importance for the further development of these approaches as practical and efficient tools for parameter identification.

In order to seek a more comprehensive validation of the applicability of the proposed (and other potential) SBO technique, further investigations considering a broader range of global biogeochemical models of different types and also the use of a full circulation model are clearly necessary. Here, our initial investigations in the context of SBO build up a crucial fundament for further research.

Acknowledgments

The authors thank Dr. Iris Kriest (GEOMAR, Kiel) for providing support to the development of the Metos3D model code, Dr. Markus Schartau (GEOMAR, Kiel) and two anonymous reviewers for their constructive and useful comments which helped to improve the manuscript. The authors further acknowledge funding to this research by the DFG Cluster of Excellence "The Future Ocean" and by the EU FP7 project "Carbo Change: Changes in carbon uptake and emissions by oceans in a changing climate".

Appendix A. Consistency conditions and SBO performance

For a well-performing SBO algorithm, the optimization of the surrogate within one iteration will lead to new parameters that bring both the surrogate and eventually the high-fidelity models closer to their optimum. More precisely, provided that the surrogate $\mathbf{s}_k$ satisfies the so-called zero- and first-order consistency conditions with the original high-fidelity model $\mathbf{y}_f(\mathbf{u}_k)$ at the current iterate $\mathbf{u}_k$, the agreement between function values and the first-order derivatives at the current iteration point as

$$\mathbf{s}_k(\mathbf{u}_k) = \mathbf{y}_f(\mathbf{u}_k), \quad \mathbf{s}'_k(\mathbf{u}_k) = \mathbf{y}'_f(\mathbf{u}_k), \tag{A.1}$$

the surrogate-based scheme in Fig. 2(b) is provable convergent to at least a local optimum of the original optimization problem (6). Furthermore, it is required that mild conditions regarding the low- and high-fidelity model smoothness are ensured and that each optimization step is enhanced by the *trust-region* (TR) safeguard, where step (3) in Fig. 2(b) is replaced by

$$\mathbf{u}_{k+1} = \operatorname{argmin}_{\substack{\mathbf{u} \in U_{ad}, \\ \|\mathbf{u}-\mathbf{u}_k\| \leqslant \delta_k}} J(\mathbf{s}_k(\mathbf{u})). \tag{A.2}$$

Employing a trust region means restricting the parameters $\mathbf{u}$ in the optimization loop to some *model-trust radius* δ_k. This radius is updated after each iteration according to the TR rules. We refer the reader to e.g. Conn et al. (2000); Koziel et al. (2010) for more details.

By definition, the surrogate proposed in this paper satisfies zero-order consistency only (cf. (11)). Formally, this is not sufficient to ensure the convergence of the surrogate-based scheme. However, as pointed out before, since the surrogate is physics-based, it inherits substantial knowledge about the high-fidelity marine ecosystem model under consideration and thus, its derivatives are expected to be at least similar to those of the high-fidelity model (cf. Section 8). Furthermore, because of being constructed from a physics-based low-fidelity model, the surrogate exhibits quite good generalization capability, which means that it provides a reliable approximation of the high-fidelity model when moving from one parameter vector to another (cf. Appendix C).

Appendix B. Surrogate model construction

B.1. Low-fidelity models

For initial experiments, we considered distinct low-fidelity models with various values of $n_{c,l}$ (cf. Fig. 1), more specifically

$$n_{c,l} = \{2000, 1600, 800, 400, 200, 100, 50, 25\}. \tag{B.1}$$

To further assess the quality of approximation of these different low-fidelity models we compare their end-of-spin-up solution with the one of the reference high-fidelity model. For this purpose, Fig. B.8 shows differences in the high- and low-fidelity model

solutions for phosphate simulated at the model's surface layer in January.

We emphasize that shown model outputs are "representative" which means that their qualitative behavior is similar for the second tracer, other points in time and depth layers.

Differences between the high- and low-fidelity model output become quite noticeable for the low-fidelity models with $n_{c,l} \leqslant 200$. This is illustrated in Fig. B.9, which shows the entire trajectories at selected locations for the corresponding model solutions. It can be seen that more or less all low-fidelity model solutions share the relevant characteristics of the high-fidelity model such as seasonal minima and maxima. Even with $n_{c,l} = 25$, the low-fidelity model solution still accounts for these main features of the high-fidelity model. We thus concentrate on the models with $n_{c,l} \leqslant 200$ in the following analysis.

B.2. Choice of a suitable low-fidelity model and correction strategy

As described in Section 6.2, the surrogate is established at each iteration k of the SBO optimization loop. For a parameter vector $\mathbf{u}_k$ which, in general, will be the result of a previous iteration, solutions of the low-fidelity model as well as the high-fidelity model are computed for $\mathbf{u}_k$ as well as for a randomly selected neighboring parameter vector $\bar{\mathbf{u}}_k$ (cf. Table B.4). Here, $\|\bar{\mathbf{u}}_k - \mathbf{u}_k\|_2 / \|\mathbf{u}_k\|_2 \approx 0.2$, i.e., $\bar{\mathbf{u}}_k$ lies in a close vicinity of $\mathbf{u}_k$. Parameter values $\mathbf{u}_k$ are taken from one iteration of an illustrative SBO run. Note that for the analysis and optimization runs presented in this paper, we did not use a possible component-wise normalization (see, e.g., Dennis and Schnabel, 1996) since parameter ranges are relatively similar.

Fig. B.10 shows the high- and low-fidelity model solutions $\mathbf{y}_f$ and $\mathbf{y}_c$ for the same illustrative tracer and spatial locations as in Fig. B.9, at the reference and neighboring point $\mathbf{u}_k$ and $\bar{\mathbf{u}}_k$. For the sake of brevity, we only show the low-fidelity model solutions with $n_{c,l} = 25$. The qualitative behavior for the other low-fidelity models, i.e., with $n_{c,l} = 50, 100, 200$ looks similar.

It can be seen that the overall "shape" of the low-fidelity model solution resembles that of the high-fidelity one. Furthermore, the qualitative relation of the high- and low-fidelity model output is rather well preserved (at least locally) for the two selected

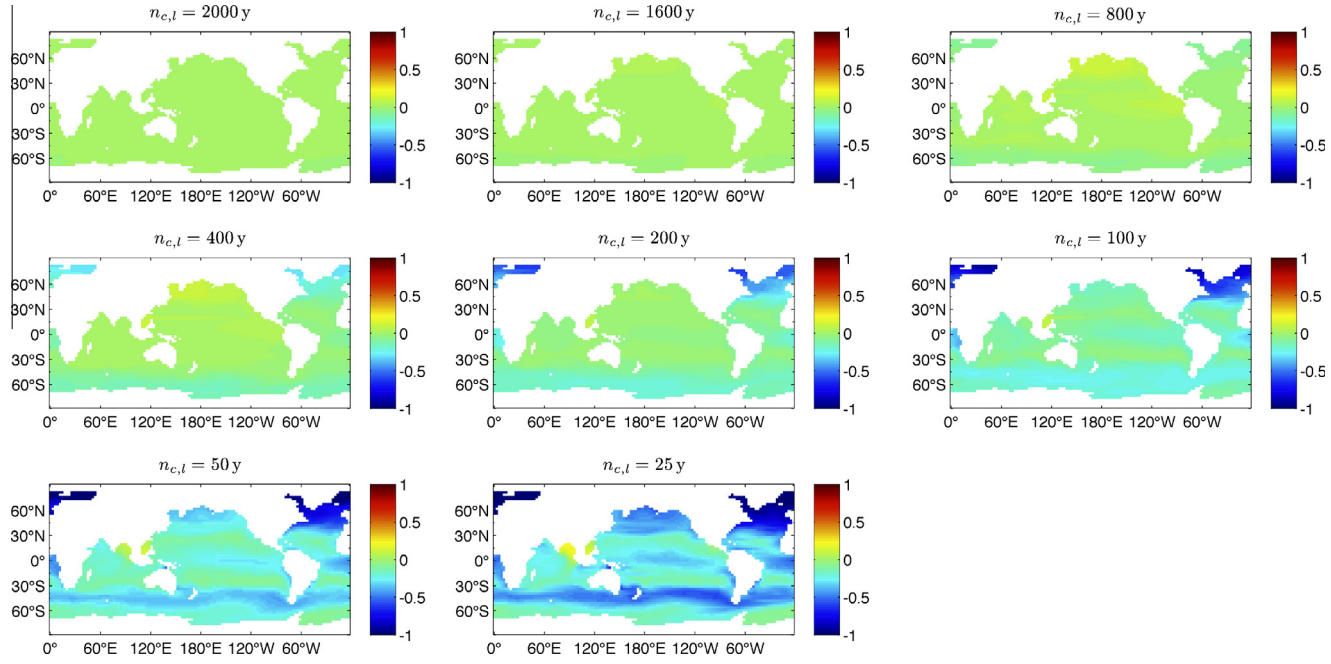
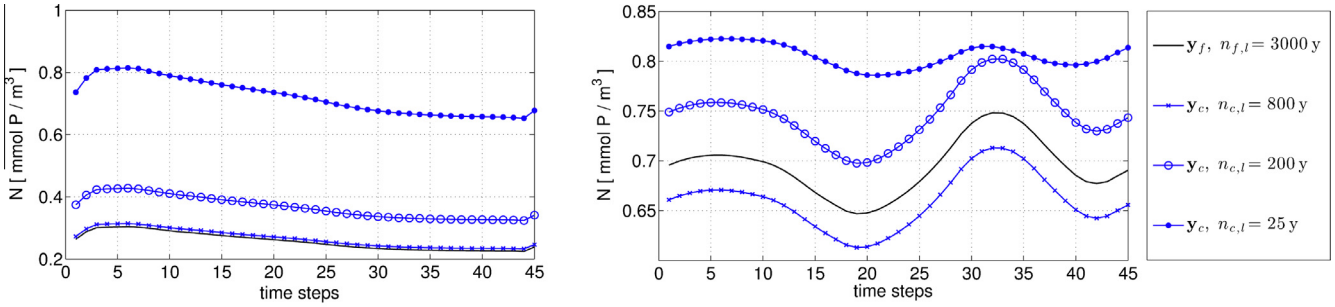


Fig. B.8. Difference in high- and low-fidelity model solutions, $\mathbf{y}_f - \mathbf{y}_c$, for illustration, here for phosphate at the model's surface layer in January. The low-fidelity models solutions are obtained by a truncated spin-up (cf. Section 6.1).

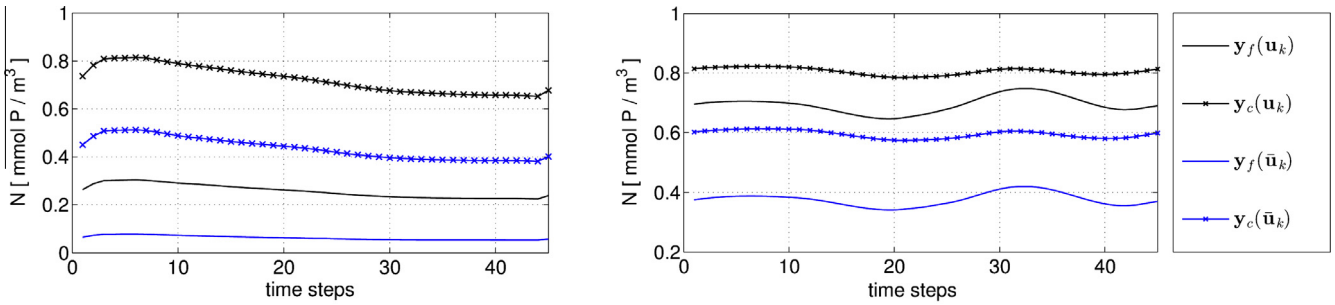


(a) $x = 30.9375°\text{W}, y = 30.9375°\text{N}$. (b) $x = 90°\text{E}, y = 0°$.

Fig. B.9. Annual cycles of surface phosphate simulated by the high- and low-fidelity models $\mathbf{y}_f, \mathbf{y}_c$, corresponding to Fig. B.8 at two different locations specified below the figure panels. For the sake of better visibility, only the low-fidelity model with $n_{c,l} = 3000, 800, 200, 25$ model years are shown.

Table B.4
Test parameter sets $\mathbf{u}_k, \bar{\mathbf{u}}_k$ and $\tilde{\mathbf{u}}_k$ to assess the quality of the surrogate model (see Figs. B.10 and C.11 and the text for details). For units see Table 1.

parameter	u_1	u_2	...				u_7
$\mathbf{u}_k$	$5 \cdot 10^{-1}$	2	$6.7 \cdot 10^{-1}$	$5 \cdot 10^{-1}$	$3 \cdot 10^{1}$	$2 \cdot 10^{-2}$	$8.58 \cdot 10^{-1}$
$\bar{\mathbf{u}}_k$	$4 \cdot 10^{-1}$	1.6	$5.36 \cdot 10^{-1}$	$4 \cdot 10^{-1}$	$2.4 \cdot 10^{1}$	$1.6 \cdot 10^{-2}$	$6.86 \cdot 10^{-1}$
$\tilde{\mathbf{u}}_k$	$4.83 \cdot 10^{-1}$	1.93	$6.48 \cdot 10^{-1}$	$4.83 \cdot 10^{-1}$	$2.9 \cdot 10^{1}$	$1.93 \cdot 10^{-2}$	$8.29 \cdot 10^{-1}$



(a) $x = 30.9375°\text{W}, y = 30.9375°\text{N}$. (b) $x = 90°\text{E}, y = 0°$.

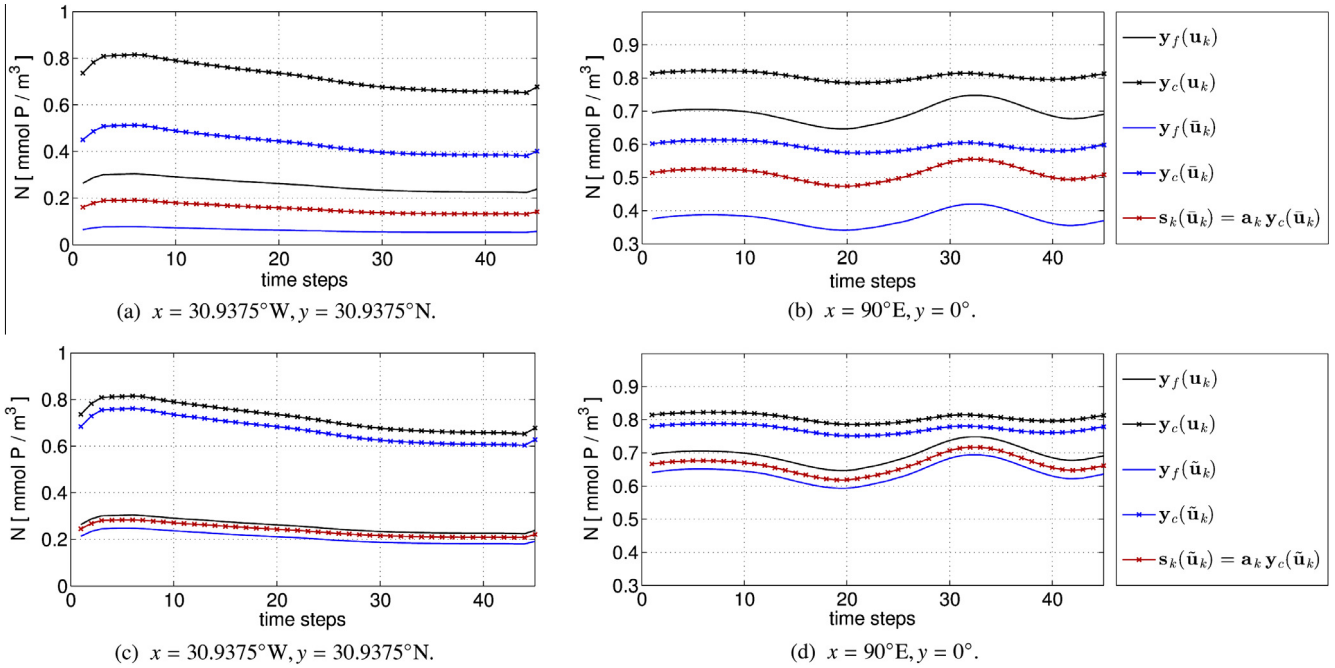
Fig. B.10. Annual cycles of surface phosphate simulated by the high- and low-fidelity models $\mathbf{y}_f, \mathbf{y}_c$ (here, for the sake of brevity, only the low-fidelity with $n_{c,l} = 25$ is shown) at the same locations as in Fig. B.9. Shown are the solutions for a reference parameter set $\mathbf{u}_k$, and a neighboring point $\bar{\mathbf{u}}_k$ (cf. Table B.4) in order assess their qualitative relation and to choose a suitable correction approach.

parameter vectors. In particular, seasonal maxima and minima occur at similar times, which is the consequence of the low-fidelity model being physics-based. The relationship between the high- and low-fidelity model solutions indicates that a *multiplicative response correction* might work well to map the low-fidelity model solution onto the high-fidelity one. This technique has already been investigated and successfully applied to a one-dimensional marine ecosystem model in Prieß et al. (2011).

Appendix C. Initial validation

Important for the performance of the SBO is the appropriateness of the surrogate also in some neighborhood of the "reference point" $\mathbf{u}_k$, i.e., the parameter vector for which the surrogate is established and for which, by definition, the model alignment is perfect (cf. (11)).

To analyze the surrogate's quality we consider the same parameter vectors $\mathbf{u}_k$ and $\bar{\mathbf{u}}_k$ with $\| \bar{\mathbf{u}}_k - \mathbf{u}_k \|_2 / \| \mathbf{u}_k \|_2 \approx 0.2$ as in Appendix



(a) $x = 30.9375°\text{W}, y = 30.9375°\text{N}$. (b) $x = 90°\text{E}, y = 0°$.

(c) $x = 30.9375°\text{W}, y = 30.9375°\text{N}$. (d) $x = 90°\text{E}, y = 0°$.

Fig. C.11. High-, low-fidelity model and surrogate's solutions $\mathbf{y}_f, \mathbf{y}_c$ and $\mathbf{s}_k$. Shown are the annual cycles of surface phosphate at two spatial locations, for one "reference point" $\mathbf{u}_k$, at a neighboring point $\bar{\mathbf{u}}_k$ (C.11a, C.11b) and in an even closer vicinity, at $\tilde{\mathbf{u}}_k$ (C.11c, C.11d), in order to assess the quality of the proposed surrogate. Parameter values are provided in Table B.4. The surrogate's solution at the reference point is omitted, since, by definition, the model alignment is perfect at this point.

B.2. Additionally, we choose another point $\tilde{\mathbf{u}}_k$ in a closer vicinity of $\mathbf{u}_k$ (cf. Table B.4), satisfying $\|\tilde{\mathbf{u}}_k - \mathbf{u}_k\|_2 / \|\mathbf{u}_k\|_2 \approx 0.03$. Fig. C.11 shows the high-, the low-fidelity and the surrogate's solutions at the reference and the neighboring points $\mathbf{u}_k, \bar{\mathbf{u}}_k$ and $\tilde{\mathbf{u}}_k$. Shown are the annual cycles for surface phosphate at two spatial locations. Solutions demonstrate that the multiplicative correction significantly increases the quality of the low-fidelity model also at the neighboring point $\bar{\mathbf{u}}_k$, whereas its accuracy is even better at the closer point $\tilde{\mathbf{u}}_k$. Fig. 3 presented in Section 8 provides additional evidence, showing the distribution of phosphate at the surface and at 455 meters depth in January.

This qualitative validation further establishes confidence that the multiplicative response correction approach in conjunction with the low-fidelity model under consideration (i.e., using $n_{c,y} = 25$) is a reasonable choice to construct a reliable surrogate. Exploiting the latter in a SBO seems very promising.

Appendix D. Trust-region convergence safeguards

At step (3) in Fig. 2(b), the design $\mathbf{u}_{k+1}$ is either accepted or rejected, depending on whether a decrease in the high-fidelity model objective function could be obtained or not. Also, the trust-region radius δ_k is updated after each iteration, i.e., decreased if the design was rejected or if the improvement of the high-fidelity model objective function was too small compared to the prediction given by the surrogate and increased otherwise. This can be briefly summarized as

$$\begin{aligned}
&\textbf{if} J(\mathbf{y}_f(\mathbf{u}_{k+1})) < J(\mathbf{y}_f(\mathbf{u}_k)) \\
&\quad \text{accept design, update(increase/decrease)}\delta_k, \\
&\quad \text{update surrogate, continue SBO} \\
&\textbf{else} \\
&\quad \text{reject design, decrease}\delta_k, \\
&\quad \text{optimize surrogate again} \\
&\textbf{endif}
\end{aligned} \tag{D.1}$$

We use classical updating rules (Conn et al., 2000; Koziel et al., 2010) with slightly modified parameters as

$$\begin{aligned}
&\delta_0 = 6 \cdot 10^{-2}, \quad \delta_{k+1} = \begin{cases} \delta_k / m_{\text{decr}}, & \text{if} \quad \rho_k < r_{\text{decr}} \\ \delta_k \cdot m_{\text{incr}}, & \text{if} \quad \rho_k > r_{\text{incr}} \end{cases}, \\
&r_{\text{incr}} = 0.75, \quad r_{\text{decr}} = 0.01, \quad m_{\text{incr}} = 3, \quad m_{\text{decr}} = 20,
\end{aligned} \tag{D.2}$$

where ρ_k denotes the *gain ratio* in iteration k defined as follows:

$$\begin{aligned}
&\rho_k := \frac{f_{\text{new}} - f_{\text{old}}}{s_{\text{new}} - s_{\text{old}}}, \\
&f_{old} := J(\mathbf{y}_f(\mathbf{u}_k)), \quad f_{new} := J(\mathbf{y}_f(\mathbf{u}_{k+1})), \\
&s_{old} := J(\mathbf{s}_k(\mathbf{u}_k)), \quad s_{new} := J(\mathbf{s}_k(\mathbf{u}_{k+1})).
\end{aligned} \tag{D.3}$$

Note that values specified above are fairly standard (see, e.g., Conn et al., 2000) except r_{decr} and m_{decr}; the first one is smaller than usual, whereas the latter is larger than usual. It was found that using this kind of setup is typically more suitable for surrogate-based optimization schemes working with physics-based surrogates (see, e.g., Koziel et al., 2013).

The surrogate defined in (8) does not satisfy the first-order consistency condition in (A.1) exactly (cf. (11)). Still, applying a trust-region safeguard appears reasonable, because the physics-based surrogate inherits substantial knowledge about the marine model under consideration so that its derivatives are expected to be at least similar to those of the high-fidelity model (see also Section 8 and Table 3). Moreover, the accuracy of the surrogate model increases with decreasing trust-region radius as indicated in Appendix C. Numerical results of an illustrative SBO run provided in Section 8 further support the applicability of a trust-region safeguard, even without using high-fidelity model sensitivity data to ensure the first-order consistency exactly.

References

Arhonditsis, G., Brett, M., 2004. Evaluation of the current state of mechanistic aquatic biogeochemical modeling. Marine Ecology Progress Series 271, 13–26.

Bandler, J., Cheng, Q., Dakroury, S., Mohamed, A., Bakr, M., Madsen, K., Sndergaard, J., 2004. Space mapping: the state of the art. IEEE Transactions on Microwave Theory and Techniques 52, 337–361.

Conn, A.R., Gould, N.I.M., Toint, P.L., 2000. Trust-region methods. MPS-SIAM series on optimization. Society for Industrial and Applied Mathematics, Philadelphia.

Dennis, J., Schnabel, R., 1996. Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Society for Industrial Mathematics.

Dutkiewicz, S., Follows, M., Parekh, P., 2005. Interactions of the iron and phosphorus cycles: A three-dimensional model study. Global Biogeochemical Cycles 19, 1–22.

Forrester, A., Keane, A., 2009. Recent advances in surrogate-based optimization. Progress in Aerospace Sciences 45, 50–79.

Khatiwala, S., 2008. Fast spin up of ocean biogeochemical models using matrix-free Newton–Krylov. Ocean Modelling 23, 121–129.

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

Koziel, S., Bandler, J.W., Cheng, Q.S., 2010. Robust trust-region space-mapping algorithms for microwave design optimization. IEEE Transactions on Microwave Theory and Techniques 58, 2166–2174.

Koziel, S., Leifsson, L., Couckuyt, I., Dhaene, T., 2013. Robust variable-fidelity optimization of microwave filters using co-kriging and trust regions. Microwave and Optical Technology Letters 55, 765–769.

Kriest, I., Khatiwala, S., Oschlies, A., 2010. Towards an assessment of simple global marine biogeochemical models of different complexity. Progress In Oceanography 86, 337–360.

Kriest, I., Oschlies, A., Khatiwala, S., 2012. Sensitivity analysis of simple global marine biogeochemical models. Global Biogeochemical Cycles 26, GB2029.

Kwon, E., Primeau, F., 2006. Optimization and sensitivity study of a biogeochemistry ocean model using an implicit solver and in situ phosphate data. Global Biogeochemical Cycles 20.

Kwon, E., Primeau, F., 2008. Optimization and sensitivity of a global biogeochemistry ocean model using combined in situ dic, alkalinity, and phosphate data. Journal of Geophysical Research: Oceans 113.

Kwon, E.Y., Primeau, F., Sarmiento, J.L., 2009. The impact of remineralization depth on the air-sea carbon balance. Nature Geoscience 2, 630–635.

Leifsson, L., Koziel, S., 2010. Multi-fidelity design optimization of transonic airfoils using physics-based surrogate modeling and shape-preserving response prediction. Journal of Computational Science 1, 98–106.

Martin, J.H., Knauer, G.A., Karl, D.M., Broenkow, W.W., 1987. Vertex: carbon cycling in the northeast pacific. Deep Sea Research Part A. Oceanographic Research Papers 34, 267–285.

Mattern, J.P., Fennel, K., Dowd, M., 2012. Estimating time-dependent parameters for a biological ocean model using an emulator approach. Journal of Marine Systems 96–97, 32–47.

Oschlies, A., 2004. Feedbacks of biotically induced radiative heating on upper-ocean heat budget, circulation, and biological production in a coupled ecosystem-circulation model. Journal of Geophysical Research – Oceans 109, C12031.

Oschlies, A., 2006. On the use of data assimilation in biogeochemical modelling. In: Chassignet, E., Verron, J. (Eds.), Ocean Weather Forecasting. Springer, Dordrecht, pp. 525–547.

Paltridge, G.W., Platt, C.M.R., 1976. Radiative Processes in Meteorology and Climatology. Elsevier, New York.

Parekh, P., Follows, M.J., Boyle, E.A., 2005. Decoupling of iron and phosphate in the global ocean. Global Biogeochemical Cycles 19, GB2020.

Parekh, P., Follows, M.J., Dutkiewicz, S., Ito, T., 2006. Physical and biological regulation of the soft tissue carbon pump. Paleoceanography 21, PA3001.

Piwonski, J., Slawig, T., 2010. The Idea and Concept of Metos3D - A Marine Ecosystem Toolkit for Optimization and Simulation in 3-D. Technical Report 1060. Christian-Albrechts-Universität zu Kiel, Institute for Computer Science.

Piwonski, J., Slawig, T., 2011. Metos3D: A Marine Ecosystem Toolkit for Optimization and Simulation. CAU Kiel, Institut für Informatik. <http://www.informatik.uni-kiel.de/co2/software/metos3d>

Prieß, M., Koziel, S., Slawig, T., 2011. Surrogate-based optimization of climate model parameters using response correction. Journal of Computational Science 2, 335–344.

Prieß, M., Koziel, S., Slawig, T., 2013a. Marine ecosystem model calibration with real data using enhanced surrogate-based optimization. Journal of Computational Science, in press.

Prieß, M., Koziel, S., Slawig, T., 2013b. Marine ecosystem model calibration through enhanced surrogate-based optimization. In: Simulation and Modeling Methodologies, Technologies and Applications. In: Pina, N., Kacprzyk, J., Filipe, J. (Eds.), . Advances in Intelligent Systems and Computing, Vol. 197. Springer, Berlin, Heidelberg, pp. 193–208.

Queipo, N.V., Haftka, R.T., Shyy, W., Goel, T., Vaidyanathan, R., Tucker, P.K., 2005. Surrogate-based analysis and optimization. Progress in Aerospace Sciences 41, 1–28.

Rückelt, J., Sauerland, V., Slawig, T., Srivastav, A., Ward, B., Patvardhan, C., 2010. Parameter optimization and uncertainty analysis in a model of oceanic CO_2-

uptake using a hybrid algorithm and algorithmic differentiation. Nonlinear Analysis: Real World Applications 10, 3993–4009.
Simpson, T.W., Poplinski, J.D., Koch, P.N., Allen, J.K., 2001. Metamodels for computer-based engineering design: Survey and recommendations. Engineering with Computers 17, 129–150.
Smola, A.J., Schölkopf, B., 2004. A tutorial on support vector regression. Statistics and Computing 14, 199–222.
Sndergaard, J., 2003. Optimization using surrogate models - by the space mapping technique. Ph.D. thesis. Informatics and Mathematical Modelling, Technical University of Denmark, DTU.
Yamanaka, Y., Tajika, E., 1997. Role of dissolved organic matter in the marine biogeochemical cycle: Studies using an ocean biogeochemical general circulation model. Global Biogeochemical Cycles 11, 599–612.

Manuscript prepared for Geosci. Model Dev.
with version 2014/09/16 7.15 Copernicus papers of the LaTeX class copernicus.cls.
Date: 17 January 2015

Metos3D: A Marine Ecosystem Toolkit for Optimization and Simulation in 3-D – Simulation Package –

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, Christian-Albrechts Universität zu Kiel, 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 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 simulation in 3-D. In addition to the usage of PETSc's parallel data structures and PETSc's Newton solver, an own load balancing algorithm is implemented. Initial experiments validate the new implementation and reveal a good parallel performance. Further model evaluations show robustness of the Newton solver with respect to parameter variations. Twin experiments confirm a good fit of the introduced interface into an optimization context. However, we found out, both solving approaches have their own specific difficulties with regard to a derivative based "black-box" optimization approach. From neither of the two we obtain satisfactory results concerning a parameter identification.

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 and assessed regarding their ability to reproduce the real world system. This involves a professional discussion of simulation results and, preferably, an estimation of optimal model parameters beforehand (cf. Fennel et al., 2001; Schartau and Oschlies, 2003).

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 high, even at lower resolution, considering optimization methods that may require hundreds of model evaluations. 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 software, namely the so-called off-line simulation, the usage of Newton's method for annual cycles and parallelization.

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 annual cycle is obtained by solving a time dependent, periodic system of nonlinear partial differential equations. The 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. This fixed point is a zero of an equivalent nonlinear residual as well (cf. Kelley, 2003). In that case, Newton-type methods 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 in 3-D is still subject to high performance computing. A 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. Except for the latter, an adequate basis for an implementation is made 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. Thus, we define a general programming 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. 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. This includes data preparation, result parsing and visualization scripts.

The remainder of this paper is organized as follows. In Sections 2–4 we describe the marine ecosystem dynamics, shortly recapitulate the transport matrix approach and define the biogeochemical model interface. In Sections 5–7 we discuss periodic solutions, go into details of the implementation and present results. Finally, Section 8 concludes our work.

2 Marine ecosystem dynamics

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

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

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 vector of all tracers. Here, we neglect the additional dependency on the time and space coordinates (t, x) in the notation for brevity.

The transport of tracers in marine waters is depicted by a diffusion and an advection term. The diffusion mixing coefficient $\kappa : I \times \Omega \to \mathbb{R}$ and the advection velocity field $v : I \times \Omega \to \mathbb{R}^3$ are regarded as given (cf. Section 3). Note that both operators effect each tracer separately. In contrast, a single component of the biogeochemical model q_i may generally depend on all tracers, i.e.

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

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 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 salinity or temperature of the ocean water. As mentioned in the introduction, the model also includes parameters that are optionally subject to optimization (cf. Table 2 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.

Additionally, homogeneous Neumann boundary conditions on the entire Γ for all tracers y_i are imposed. 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. 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. Accordingly, we solve the model equations by computing an annual cycle, which is a vector of tracer concentrations with $y(t+1, x) = y(t, x)$ (cf. Section 5).

3 Transport matrices

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

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

where $L(t)$ comprises both and represents a time dependent linear operator. Formally, its fully discrete equivalent is a sequence of matrices $(\mathbf{L}_j)_{j=1}^{n_t}$ with $\mathbf{L}_j = \mathbf{L}(t_j)$. 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 $\Delta t = 1/n_t$. 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. 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). This requires code knowledge and implies a technical effort that is described by Khatiwala et al. (2005) for instance. As a general rule, once the discretization parameters are chosen, the arrangement of the

transport matrices, the boundary and domain data and the tracer vectors are determined for further usage.

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

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

Here, the transport is split as $\mathbf{L}_j = \mathbf{L}_{imp,j} + \mathbf{L}_{exp,j}$ and $\mathbf{I}$ represents the identity. Throughout this work, both matrix types are sparse. The implicit matrix $\mathbf{L}_{imp,j}$ comprises vertical diffusion only, i.e. a process within a water column that is computed and inverted independently of its vicinity. The explicit matrix $\mathbf{L}_{exp,j}$ represents a (local) differential operator, which naturally has a sparse discrete representation.

Overall, the fully discrete iteration scheme for n tracers results in a block diagonal system. The integration of state variables over a model year consists of sparse matrix vector multiplications and evaluations of the biogeochemical model. For a fixed time index j it reads

$$\begin{aligned}\mathbf{y}_{j+1} &= \mathbf{A}'_{imp,j}(\mathbf{A}'_{exp,j}\,\mathbf{y}_j + \Delta t\, \mathbf{q}_j(\mathbf{y}_j, \boldsymbol{u}, \mathbf{b}_j, \mathbf{d}_j)) \\ &= \varphi_j(\mathbf{y}_j, \mathbf{u}, \mathbf{b}_j, \mathbf{d}_j)\,,\end{aligned} \tag{2}$$

where $\mathbf{y}_j = (\mathbf{y}_i(t_j))_{i=1}^n$ combines all discrete tracer vectors. 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. The components of the tracer model are depicted by $\mathbf{q}_j$ with

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

where the discrete boundary respectively domain data is represented by $\mathbf{b}_j = (\mathbf{b}_i(t_j))_{i=1}^{n_b}$ and $\mathbf{d}_j = (\mathbf{d}_i(t_j))_{i=1}^{n_d}$.

Actually, only 12 implicit and 12 explicit matrices are extracted and stored, when the TMM data is prepared. They represent monthly averaged ocean circulation, but provide a sufficient accuracy at minimal storage requirements as shown by Khatiwala et al. (2005). The same applies for the given forcing. 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).

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

Thus, 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 1). Hence, a vector length n_y is a sum of non-equidistant profiles, i.e.

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

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

The evaluation of the whole n 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

$$\Delta t\,(\mathbf{q}_i(t_j, (\mathbf{y}_i)_{i=1}^n, \mathbf{u}, (\mathbf{b}_i)_{i=1}^{n_b}, (\mathbf{d}_i)_{i=1}^{n_d}))_{i=1}^n\,. \tag{3}$$

Here, $(\mathbf{y}_i)_{i=1}^n$ is an input array of n profiles, $\mathbf{u}$ a vector of m parameters, $(\mathbf{b}_i)_{i=1}^{n_b}$ a vector of n_b boundary data values and $(\mathbf{d}_i)_{i=1}^{n_d}$ 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 $(\mathbf{q}_i)_{i=1}^n$ that consist of n 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_b and n_d. They are followed by `real*8` (double precision) arguments, i.e. Δt, $\mathbf{q}$, t_j, $\mathbf{y}$, $\mathbf{u}$, $\mathbf{b}$ and $\mathbf{d}$. 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`.

5 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 (1) 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

$$\phi(\mathbf{y}_1, \mathbf{u}) = \mathbf{y}_1, \tag{4}$$

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 (2)). 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).

5.2 Inexact Jacobian-free Newton-Krylov

On the other hand, Equation (4) 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

$$F'(\mathbf{y}_k)\,\mathbf{s}_k = -F(\mathbf{y}_k)\,, \tag{5}$$

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 (5) 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.

5.3 Convergence

Assuming there exist a unique solution of Equation (1), 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

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

5.3.1 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.

5.3.2 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 oversolving and decreases, especially in the beginning, the number of evaluations of F. The scaling factor η_k is determined from former Newton residuals as

$$\eta_k = \gamma \left(\frac{\|F(\mathbf{y}_k)\|_2}{\|F(\mathbf{y}_{k-1})\|_2} \right)^{\alpha} \tag{6}$$

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`, `data`, `model` and `simpack`. The first comprises the installation scripts, the second the model source codes and the third all the data preparation scripts as well as the data. The latter consist of 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 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. 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 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 initialization respectively finalization routines are combined as one call within the *init* source file.

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 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 $((\mathbf{d}_{i,j})_{j=1}^{n_{d,i}})_{i=1}^{n_d}$. 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 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 $(\mathbf{b}_{i,j})_{j=1}^{n_{b,i}}$ we compute the appropriate weights α, β as well as indices j_α, j_β and form the linear combination as

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

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

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

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.

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

where $\mathbf{y}_{i,k}$ 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.

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

where $\mathbf{d}_{i,k}$ 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 using the interface 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 $\mathbf{y}_{in}$.

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 validate the software. At first, 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 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 validation.

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 Model

In order to test our interface, we decide to couple an *original* implementation of a biogeochemical model that 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 2). 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 convention for the directory structure to fit seamlessly into the automatic compile scheme. Within the `model` directory of the `model` repository we create a folder named `MITgcm-PO4-DOP`. We implement a model wrapper for the original source code and store it in a file named `model.F` within that folder. 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
```

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 1). The depth is divided into 15 vertical layers that are depicted in Table 1. This geometry translates to a (single) tracer vector length of $n_y = 52749$ 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 2). The components of the combined tracer vector are $\mathbf{y}_{\mathrm{PO4}}$ and accordingly $\mathbf{y}_{\mathrm{DOP}}$, i.e. $\mathbf{y} = (\mathbf{y}_{\mathrm{PO4}}, \mathbf{y}_{\mathrm{DOP}})$. The photosynthetically available short wave radiation 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, 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 $\mathbf{y}_{0,\,\mathrm{PO4}} = 2.17\ \mathrm{mmol\,P/m^3}$ and $\mathbf{y}_{0,\,\mathrm{DOP}} = 0.0001\ \mathrm{mmol\,P/m^3}$, respectively.

7.2 Solver

We begin our validation by computing a reference solution for the parameter set $\mathbf{u}_d$ that is depicted in Table 2. 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 3000 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 2 shows the convergence towards a periodic steady state. Both solver obviously converge towards the same solution. The difference is generally measured using the unweighted norm of initial states consecutive model years. Additionally, every 100 years we computed the weighted norm between whole trajectories for comparison.

However, 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, 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 solutions presented by Kwon and Primeau (2006) for instance. Figure 3 shows the concentration of phosphate within the first layer. Here, the data is shifted to show Greenwich ($0°$) at the center. Moreover, Figure 4 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, 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 at which we iterate for 10 model years. The analysis of the profiling results is shown in Figure 5. 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%.

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.

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 second is an Intel® Sandy Bridge EP architecture with Intel Xeon® E5-2670 CPUs that consist of 8 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, 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 of three model years, at which each year is timed separately. 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 timings are related to 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 $N = 184$ 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, 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 6 depicts the ideal, best possible and actual speedup respectively efficiency. Regarding the implemented load distribution a good performance over the whole range of processes can be observed. However, 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 on the Intel hardware reaches between 100 and 140 processes 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 ar-

chitecture. Indeed, the speed-up still rises to slightly over 160 but requires at least 200 processes to reach this factor.

Interestingly, there is a significant drop in performance at the beginning on both architectures. 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 validation of results and a review of technical aspects of our implementation, we investigate the settings to control the convergence of the Newton-Krylov solver. Our intention is to eliminate the over-solving that we observe during the first 200 iterations in Figure 2. 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 (6).

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 3000 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 7 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.

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 validation, 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 now we solved the given model equations for the reference parameter set $\mathbf{u}_d$ 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. 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 2. We set the overall tolerance again to a value that is comparable with 3000 spin-up iterations and let the Newton solver compute a solution for each parameter sample

Figure 8 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

Finally, after a validation 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. Regarding the spin-up we let the solver iterate for 3000 model years and set no tolerance once again. The Newton solver is set up as described in Section 7.5.

We consider the following optimization problem:

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

where

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

and the admissible set is defined as

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

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. The norm is computed using the whole trajectory and both optimization runs are started with $\mathbf{u}_0$ (cf. Table 2).

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). It is a quasi-Newton approach, at which the inverse of the Hessian is approximated using Broyden's method (cf. Dennis and Schnabel, 1996, pp. 169). 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. Regarding the Newton solver, one additional experiment is carried out with a relative step size of 10^{-4}.

Figure 9 shows the results of the optimization run using the spin-up solver. We observe that the optimizer do not finish its search for a minimum. Though we use 128 processors, the computation exceedes the queue limit of 200 hours of the used batch processing system. However, we recognize a decay of the cost function and a tendency of the parameters towards their reference values. At the last optimization step they are $\mathbf{u} = (0.499, 2.209, 0.680, 0.546, 34.922, 0.019, 0.870)$. Here, the values are round off to three decimal places.

Figure 10 depicts the attempt to minimize the cost function using the Newton solver for model evaluation. Both optimization runs finish because the predicted change in the objective function is less than 10^{-6}, which is 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. Here, based on the two experiments, a detailed analysis is hardly possible. They provide only first clues (cf. Section 8).

8 Conclusions

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 models to 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 validation. 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. Here, we have seen how useful the inherited profiling capability can be to access the computational complexity of a new model implementation. 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. The PETSc library provides a flexible and robust solver implementation that, in our case, solves the given nonlinear equations at least 6 times faster than the fixed point iteration. 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. Note that the chosen optimization approach was somehow "natural". 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. 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 510.000 model years were simulated during the spin-up twin experiment and, after all, we used 128 processes for about 200 hours just to realize it was not sufficient. The approach may be easy to realize, but it clearly consumes to many computational resources. 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. Indeed, the appropriate scripts are provided in the `data` repository of Metos3D, but once again, not to further complicate a basic validation the approach was not discussed here.

Moreover, due to the fact that a coarser time step many lead to inaccurate results, a Newton solver was integrated into the software. And, as it turned out, a model evaluation using a Newton approach is much faster. 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.

Furthermore, usually the employment of pre-conditioners must be taken into account, as has already been discussed by Khatiwala (2008). Indeed, PETSc offers several own preconditioner implementations or at least the possibility to interact with the inner solver at the appropriate location. However, none of the included PETSc pre-conditioner nor the presented approach by Khatiwala (2008) is matrix-free. Thus, once again, in order to not further complicate the basic validation this has not been considered here.

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.

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. 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, 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, 2012a.

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

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

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

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, 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, 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.

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, 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, GB2029, http://oceanrep.geomar.de/14320/, 2012.

Kwon, E. and Primeau, F.: Optimization and sensitivity study of a biogeochemistry ocean model using an implicit solver and in situ phosphate data, Global Biogeochem. Cycles, 20, 2006.

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, 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.

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, 1999.

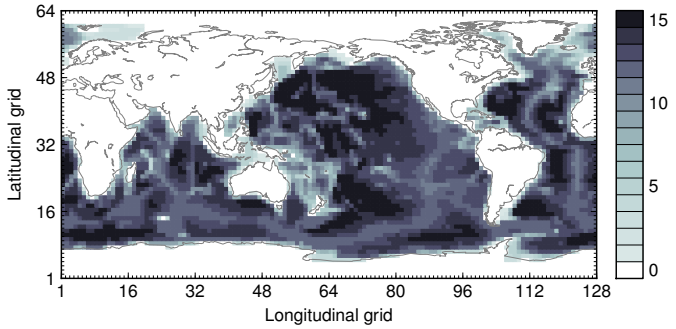


Figure 1. Land-sea mask (geometric data) of the used numerical model. Shown are the number of layers per grip point.

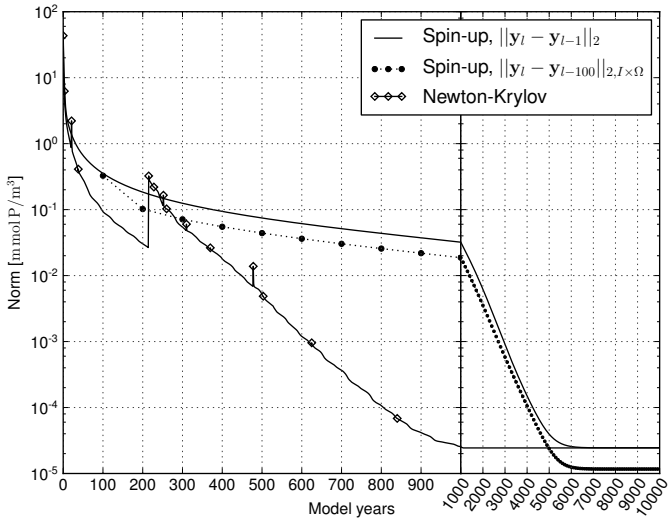


Figure 2. Convergence towards an annual cycle. *Spin-up:* norm of difference between initial states of consecutive model years (solid line) and trajectories every hundred model years (dots with dashed line). *Newton-Krylov:* residual norm at a Newton step (diamond) and norm of the GMRES residual during solving (solid line in-between).

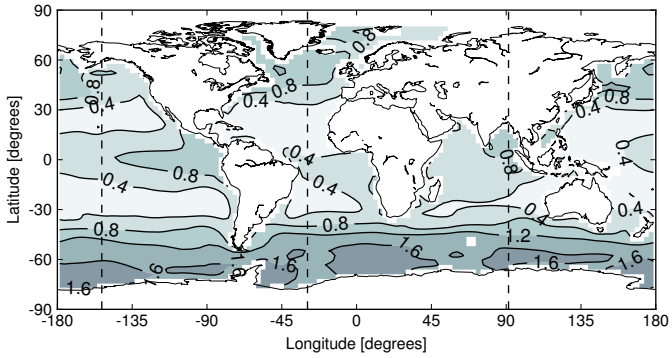


Figure 3. Concentration of phosphate ($\mathbf{y}_{PO4}$) at the first layer (0 – 50 m). Shown is the initial state (1st of January, 00:00 am) of the converged annual cycle presented in Figure 2. The dashed lines depict locations of slices shown in Figure 4.

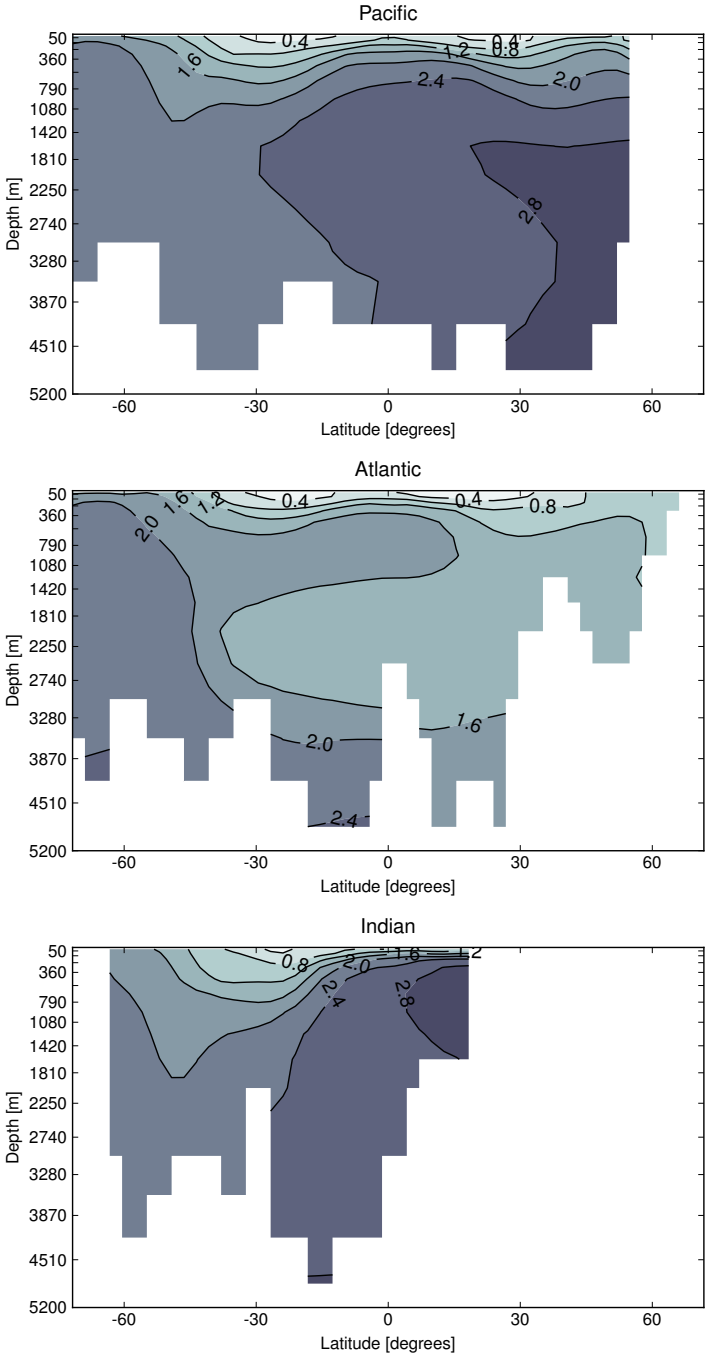


Figure 4. Slices corresponding to Figure 3: the Pacific (153.2815° W), the Atlantic (29.53125° W) and the Indian (91.40625° E).

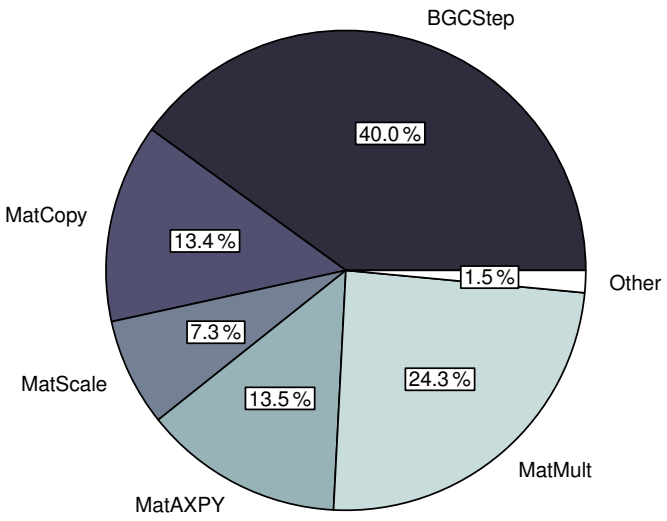


Figure 5. Distribution of the computational time among main operations during the integration of a model year.

Table 1. 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

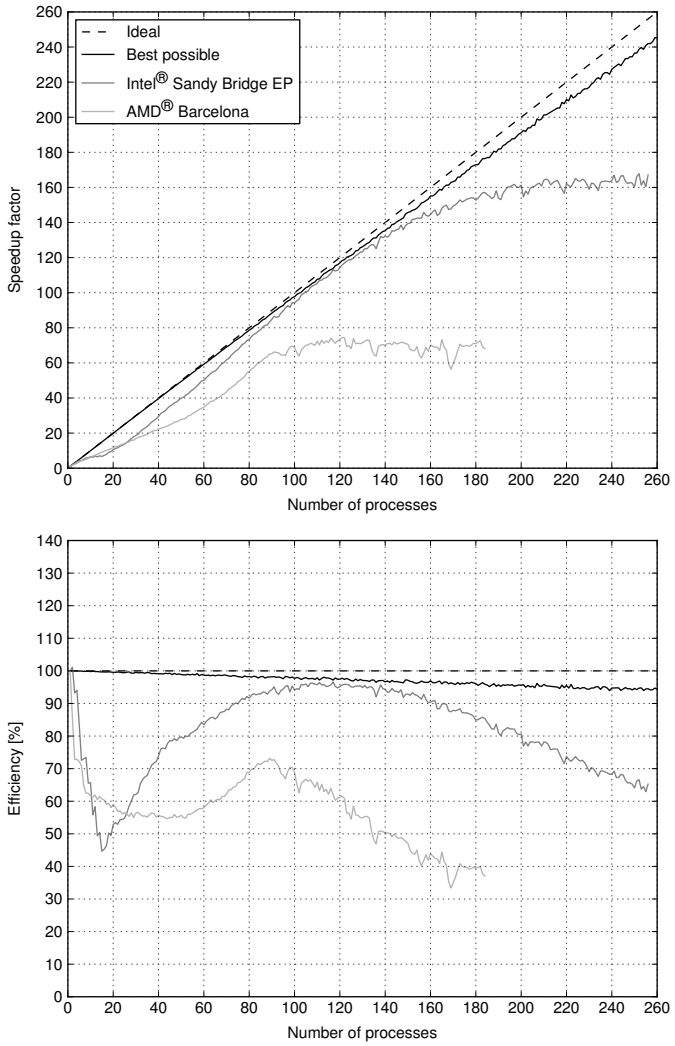


Figure 6. Ideal and actual speedup factor as well as efficiency of parallelized computations. Here, best possible refers to the used load distribution introduced in Section 7.4.

Table 2. Parameters implemented in the MITgcm-PO4-DOP model. Specified are the location within the parameter vector, the description of the parameter 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 as well as the initial parameter guess ($\mathbf{u}_0$) used during the twin experiment.

$\mathbf{u}$	Description	$\mathbf{u}_d$	$\mathbf{b}_l$	$\mathbf{u}_0$	$\mathbf{b}_u$	Unit
u_1	DOP remineralization rate	0.5	0.25	0.3	0.75	$1/\mathrm{y}$
u_2	maximum community production	2.0	1.5	5.0	200.0	$1/\mathrm{y}$
u_3	fraction of DOP	0.67	0.05	0.4	0.95	1
u_4	PO4 half saturation	0.5	0.25	0.8	1.5	$\mathrm{m\,mol\,P/m^3}$
u_5	light half saturation	30.0	10.0	25.0	50.0	$\mathrm{W/m^2}$
u_6	light attenuation	0.02	0.01	0.04	0.05	$1/\mathrm{m}$
u_7	power law remineralization coefficient	0.858	0.7	0.78	1.5	1

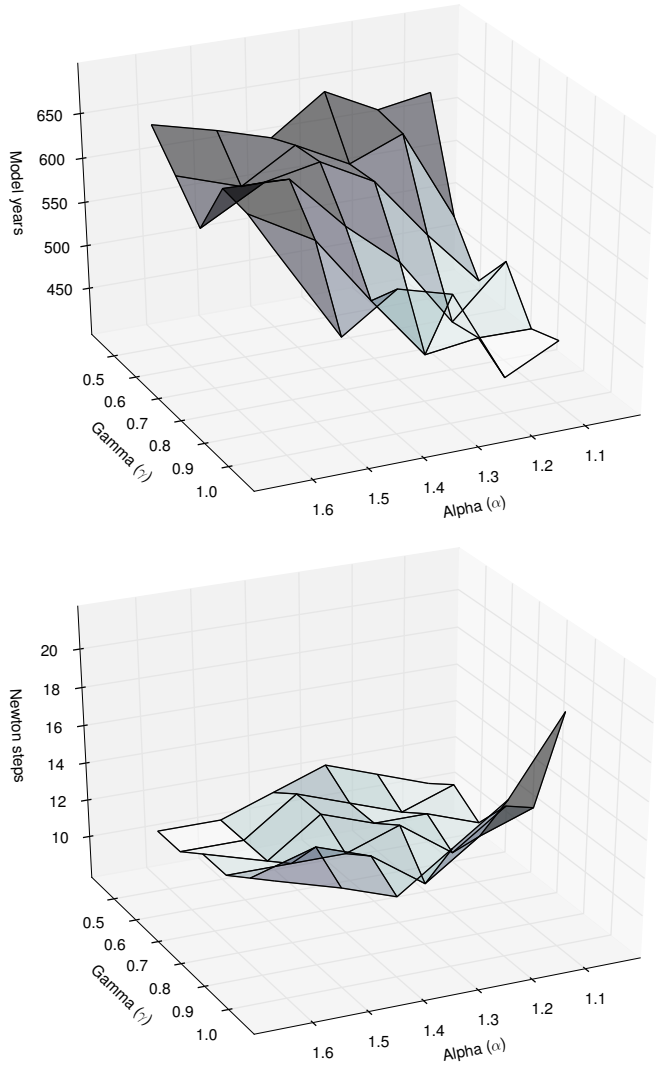


Figure 7. Number of model years and Newton steps required for the computation of the annual cycle $\mathbf{y}(\mathbf{u}_d)$ as a function of different convergence control parameters α and γ (cf. Equation (6)).

Algorithm 1: Load balancing

Input : vector length: n_y, number of profiles: n_p, profile lengths: $(n_{y,k})_{k=1}^{n_p}$, 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**
    $i = \text{floor}(((w + 0.5 * n_{y,k})/n_y) * N)$;
    $n_{p,i} = n_{p,i} + 1$;
    $w = w + n_{y,k}$;
end

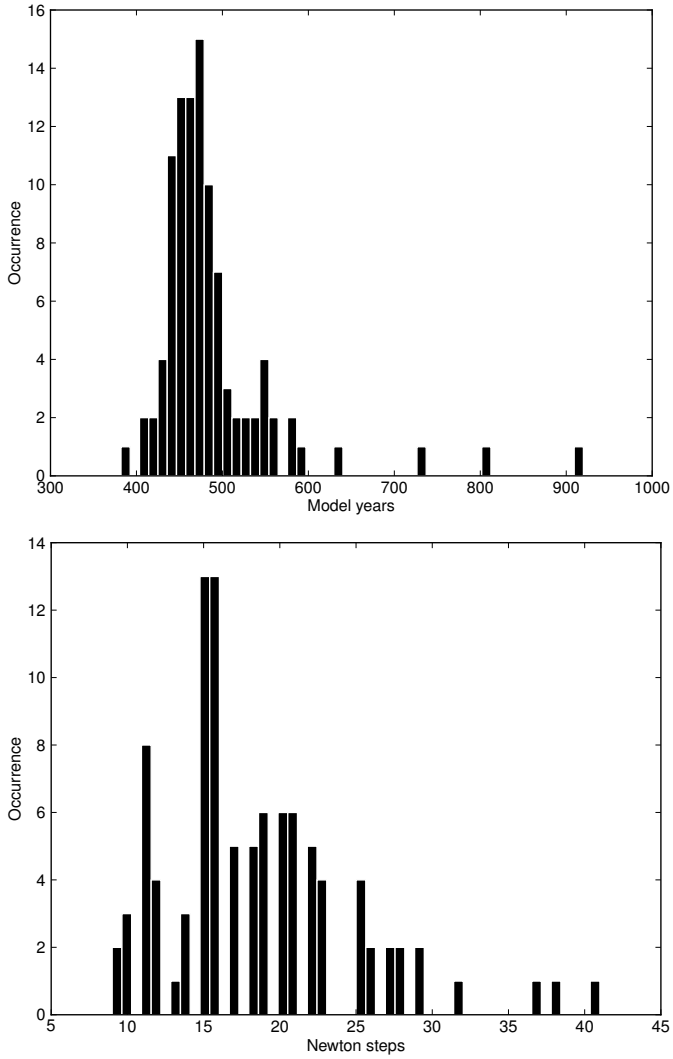


Figure 8. Distribution of number of model years and Newton steps required for the computation of a annual cycle using 100 random parameter samples (cf. Section 7.6).

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

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

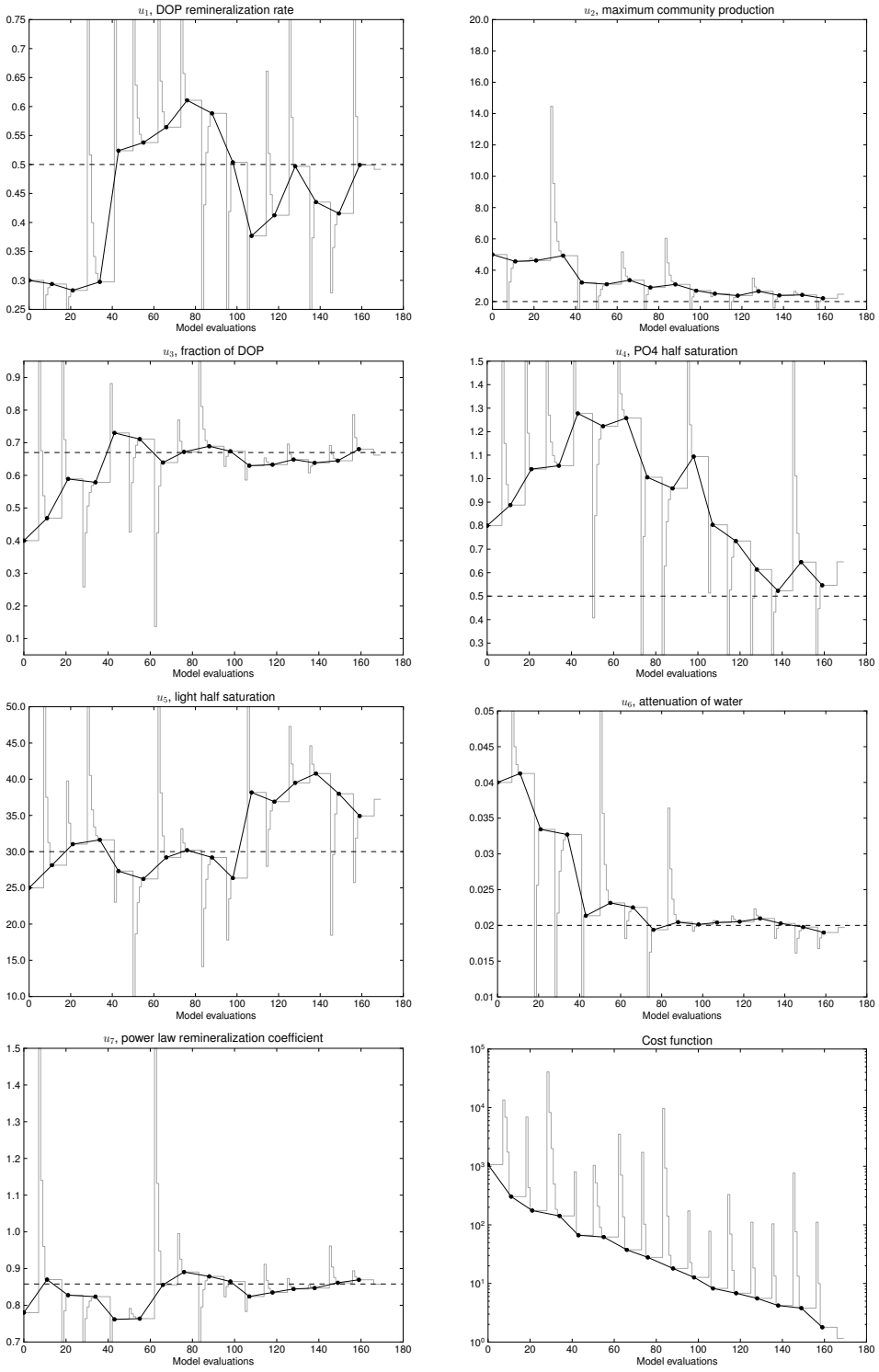


Figure 9. Twin experiment using the spin-up solver. Bullets on the black line depict the steps of the optimization process. The gray line shows all model evaluations including gradient computation and line search step. Vertical limits of the figures are also parameter bounds (except cost function and second parameter).

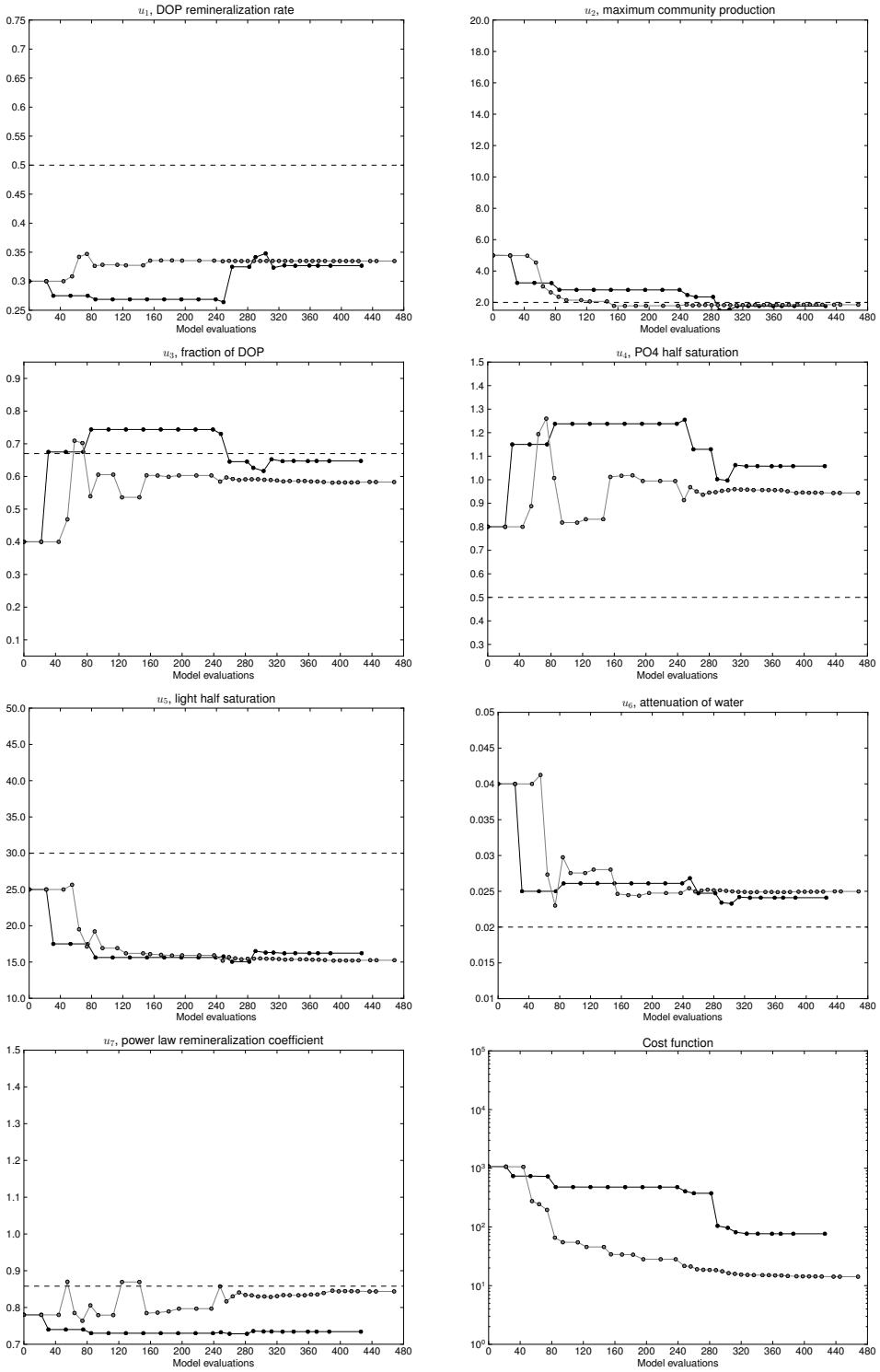


Figure 10. Twin experiment using the Newton solver. The black line depicts the steps taken by the optmizer using a absolute finite difference step that equals to the square root of the machine precision. The gray line refers to a relative finite difference step of 10^{-4}. Intermediate model evaluations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

Algorithm 3: `Phi` (ϕ)

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

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

$\mathbf{y}_{in} = \mathbf{y}_0$;
for $j = 1, \ldots, n_t$ **do**
    $t_j = \mod(t_0 + (j-1)\Delta t, 1.0)$;
    $\mathbf{y}_{out} = \texttt{PhiStep}(t_j, \Delta t, \mathbf{A}_{imp}, \mathbf{A}_{exp}, \mathbf{y}_{in}, \mathbf{u}, \mathbf{b}, \mathbf{d})$;
    $\mathbf{y}_{in} = \mathbf{y}_{out}$;
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: $\mathbf{y}_{in}$, parameters: $\mathbf{u} \in \mathbb{R}^m$, boundary data: $\mathbf{b}$, domain data: $\mathbf{d}$

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

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

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

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

BIBLIOGRAPHY

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, 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, 2012.

Banks, H. T. and Kunisch, K.: Estimation Techniques for Distributed Parameter Systems, Birkhäuser, 1989.

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.

Borzì, A. and Schulz, V.: Computational optimization of systems governed by partial differential equations, vol. 8, SIAM, 2012.

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, 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 Hamdi, A.: Properties of an Augmented Lagrangian for Design Optimization, Optimization Methods and Software, 25, 645–664, 2010.

Griewank, A. and Hamdi, A.: Reduced Quasi-Newton Method for Simultaneous Design and Optimization, Computational Optimization and Applications Online, 49, 521–548, 2011.

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

Herzog, R.: Algorithms and Preconditioning in PDE-Constrained Optimization, Summer school Lectures Notes, Lambrecht, 2010.

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

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.

Kwon, E. and Primeau, F.: Optimization and sensitivity study of a biogeochemistry ocean model using an implicit solver and in situ phosphate data, Global Biogeochem. Cycles, 20, 2006.

Munson, T., Sarich, J., Wild, S., Benson, S., and McInnes, L. C.: TAO 2.0 Users Manual, Tech. Rep. ANL/MCS-TM-322, Mathematics and Computer Science Division, Argonne National Laboratory, http://www.mcs.anl.gov/tao, 2012.

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

Piwonski, J. and Slawig, T.: Metos3D: A Marine Ecosystem Toolkit for Optimization and Simulation in 3-D – Simulation Package –, submitted to Geoscientific Model Development Discussions, 2015.

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

Prieß, M., Piwonski, J., Koziel, S., and Slawig, T.: Parameter Identification in Climate Models using Surrogate-based Optimization, Journal of Computational Methods in Science and Engineering, 12, 47–62, doi:10.3233/JCM-2012-0403, 2012.

Prieß, M., Piwonski, J., Koziel, S., Oschlies, A., and Slawig, T.: Accelerated parameter identification in a 3D marine biogeochemical model using surrogate-based optimization, Ocean Modelling, 68, 22 – 36, doi:10.1016/j.ocemod.2013.04.003, 2013.

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.

Stocker, T., Qin, D., Plattner, G.-K., Tignor, M., Allen, S., Boschung, J., Nauels, A., Xia, Y., Bex, V., and Midgley, P., eds.: IPCC, 2013: Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change, Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA, 2013.

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

Acknowledgements

First of all, I would like to thank Prof. Dr. Thomas Slawig for his exemplary supervision. He believed in me, he encouraged me and, to put it in a nutshell, he enabled this work at all. Thank you, Thomas, it was an exciting and instructive time! Next, I would like to thank Prof. Dr. Andreas Oschlies for the great collaboration. He made an interdisciplinary research possible. Thank you, Andreas, I am looking forward to a sequel! I also would like to thank Dr. Iris Kriest for patiently teaching me mostly everything I know about biogeochemical models. Thanks, Iris, I enjoyed our discussions! I would further like to acknowledge proofreading by Claudia Kratzenstein and Joscha Reimer as well as funding to this research by the DFG Cluster of Excellence "The Future Ocean".

Moreover, I want to say thanks to my parents for their patience and financial support. Thanks, Jana and Heinz! Most important of all, I want to thank my wife Nina for backing me up and making my life worthwhile. Thank you so very much, Nina! You made me happy the first time I saw you and you still do. Last but not least, I want to acknowledge all the love I received from my daughter Jule during the stressful time. Thanks, sweety!

Erklärung

Hiermit versichere ich,

I. dass diese Arbeit – abgesehen von der Beratung durch die Betreuer Thomas Slawig und Andreas Oschlies – nach Inhalt und Form meine eigene ist,

II. dass diese Arbeit zum Teil an einer anderen Stelle bereits veröffentlicht, bzw. zur Veröffentlichung eingereicht wurde, im Detail:

II.a. Prieß, M., Piwonski, J., Koziel, S., and Slawig, T.: Parameter Identification in Climate Models using Surrogate-based Optimization, Journal of Computational Methods in Science and Engineering, 12, 4762, doi:10.3233/JCM-2012-0403, 2012.

II.b. Siewertsen, E., Piwonski, J., and Slawig, T.: Porting marine ecosystem model spin-up using transport matrices to GPUs, Geoscientific Model Development, 6, 1728, doi:10.5194/gmd-6-17-2013, 2013.

II.c. Prieß, M., Piwonski, J., Koziel, S., Oschlies, A., and Slawig, T.: Accelerated parameter identification in a 3D marine biogeochemical model using surrogate-based optimization, Ocean Modelling, 68, 22 36, doi:10.1016/j.ocemod.2013. 04.003, 2013.

II.d. Piwonski, J., and Slawig, T.: Metos3D: A Marine Ecosystem Toolkit for Optimization and Simulation in 3-D - Simulation Package -, Geoscientific Model Development Discussions, **eingereicht am 08.01.2015**

III. dass diese Arbeit unter Einhaltung der Regeln guter wissenschaftlicher Praxis der Deutschen Forschungsgemeinschaft entstanden ist,

IV. und dass die eingereichten Veröffentlichungen einen Eigenanteil gemäß des Schreibens meines Betreuers vorweisen.

Jaroslaw Piwonski
21. April 2015