Geosci. Model Dev. Discuss., 8, 4401–4451, 2015
www.geosci-model-dev-discuss.net/8/4401/2015/
doi:10.5194/gmdd-8-4401-2015

Geoscientific Model Development Discussions
Open Access

Metos3D: a marine ecosystem toolkit for optimization and simulation in 3-D – Simulation Package v0.2

J. Piwonski and T. Slawig

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

Received: 8 January 2015 – Accepted: 13 May 2015 – Published: 16 June 2015

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

Published by Copernicus Publications on behalf of the European Geosciences Union.

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 off-line 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.

A simple verification is carried out using a well investigated biogeochemical model for phosphate (PO_4) and dissolved organic phosphorous (DOP) with 7 parameters. The model is coupled via the interface to transport matrices that correspond to a longitudinal and latitudinal resolution of 2.8125° and 15 vertical layers. Initial tests show that both solvers and the load balancing algorithm work correctly. Further experiments demonstrate the robustness of the Newton solver with respect to parameter variations. Moreover, the numerical tests reveal that, with optimal control settings, the Newton solver converges at least 6 times faster towards a solution than the spin-up.

However, additional twin experiments reveal differences between both solvers regarding a derivative-based black-box optimization. Whereas an optimization run with spin-up-based model evaluations is capable to identify model parameters of a reference solution, Newton-based model evaluations result in an inaccurate gradient approximation.

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 ecosys-

GMDD
8, 4401–4451, 2015

tems are understood as extensions of the latter (cf. Fasham, 2003; Sarmiento and Gruber, 2006). Consequently, we will use both terms synonymously below. However, whereas the equations and variables of ocean dynamics are well known, descriptions of biogeochemical or ecological sinks and sources still entail uncertainties concerning the number of components and parameterizations (cf. Kriest et al., 2010). A wide range of marine ecosystem models needs to be validated, i.e. assessed regarding their ability to 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).

GMDD
8, 4401–4451, 2015

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 Sects. 2–4 we describe the marine ecosystem dynamics, shortly recapitulate the transport matrix approach and define the biogeochemical model interface. In Sects. 5–7 we discuss periodic solutions,

GMDD
8, 4401–4451, 2015

go into details of the implementation and present results. Finally, Sect. 8 concludes our work and Sect. 9 describes how to obtain the source code.

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 the text

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

where y_i is a tracer concentrations with $y_i : I \times \Omega \to \mathbb{R}$ and $\boldsymbol{y} = (y_i)_{i=1}^{n}$ 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. Sect. 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.

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

Here, $\boldsymbol{b} = (b_i)_{i=1}^{n_\mathrm{b}}$ with $b_i : I \times \Gamma_\mathrm{s} \to \mathbb{R}$ is a vector of boundary forcing data like insolation or wind speed, which is defined on the ocean surface $\Gamma_\mathrm{s} \subset \Gamma$. Additionally, $\boldsymbol{d} = (d_i)_{i=1}^{n_\mathrm{d}}$ with $d_i : I \times \Omega \to \mathbb{R}$ is a vector of domain forcing data like salinity or temperature of the ocean water (cf. Sect. 4). As mentioned in the introduction, the model also includes parameters that are optionally subject to optimization (cf. Table 2 as an example). They

GMDD
8, 4401–4451, 2015

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 $\kappa, v, \boldsymbol{b}$ and $\boldsymbol{d}$ 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 $\boldsymbol{y}(t+1,x) = \boldsymbol{y}(t,x)$ (cf. Sect. 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

GMDD
8, 4401–4451, 2015
Metos3D
J. Piwonski and T. Slawig

the implicit time step and an explicit transport matrix as the application of the explicit time step, i.e.

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

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

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

where $\boldsymbol{y}_j = (\boldsymbol{y}_i(t_j))_{i=1}^{n}$ combines all discrete tracer vectors. Accordingly, $\mathbf{A}'_{\mathrm{imp},j}$ and $\mathbf{A}'_{\mathrm{exp},j}$ denote block diagonal matrices with $\mathbf{A}_{\mathrm{imp},j}$ and $\mathbf{A}_{\mathrm{exp},j}$ as their identical blocks, respectively. The components of the tracer model are depicted by $\boldsymbol{q}_j$ with

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

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

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

GMDD
8, 4401–4451, 2015

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

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

where n_p is the number of profiles and $(n_{y,k})_{k=1}^{n_\mathrm{p}}$ 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

GMDD
8, 4401–4451, 2015

$n_{y,k}$ we compute

$$\Delta t \left(\boldsymbol{q}_i \left(t_j, (\boldsymbol{y}_i)_{i=1}^{n}, \boldsymbol{u}, (\boldsymbol{b}_i)_{i=1}^{n_\mathrm{b}}, (\boldsymbol{d}_i)_{i=1}^{n_\mathrm{d}} \right) \right)_{i=1}^{n} . \tag{3}$$

Here, $(\boldsymbol{y}_i)_{i=1}^{n}$ is an input array of n profiles, $\boldsymbol{u}$ a vector of m parameters, $(\boldsymbol{b}_i)_{i=1}^{n_\mathrm{b}}$ a vector of n_b boundary data values and $(\boldsymbol{d}_i)_{i=1}^{n_\mathrm{d}}$ 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 $(\boldsymbol{q}_i)_{i=1}^{n}$ 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, $\boldsymbol{q}$, t_j, $\boldsymbol{y}$, $\boldsymbol{u}$, $\boldsymbol{b}$ and $\boldsymbol{d}$. We neglected the profile index k and the time index j in the notation for clarity. Moreover, we use `dt` as a textual representation of Δt.

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

GMDD
8, 4401–4451, 2015

5 Periodic solution

With those two building blocks, a model evaluation for a given parameter set $\boldsymbol{u} \in \mathbb{R}^m$ is a calculation of an annual periodic state that solves Eq. (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 $(\boldsymbol{y}_j)_{j=1}^{n_t}$ with

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

where $\phi = \varphi_{n_t} \circ \ldots \circ \varphi_1$ is the mapping that integrates a given tracer concentration over a model year (cf. Eq. 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 $\boldsymbol{y}_1$. 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.

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

where $l = 1, \ldots, n_l$ is the model year index, n_l is the overall number of model years and $\boldsymbol{y}_l$ 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, Eq. (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 pur-

GMDD
8, 4401–4451, 2015
Metos3D
J. Piwonski and T. Slawig

pose, we define $F(\boldsymbol{y}, \boldsymbol{u}) = \boldsymbol{y} - \phi(\boldsymbol{y}, \boldsymbol{u})$ and solve $F(\boldsymbol{y}, \boldsymbol{u}) = 0$ for a given parameter set $\boldsymbol{u}$. However, we omit the dependency of F on $\boldsymbol{u}$ in the following for clarity.

Using a Newton iteration in every step we solve

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

where $k = 1, \ldots$ is the Newton step index, F' denotes the Jacobian of F and $\boldsymbol{s}_k$ is the state update to find that is used to form the next iterate, i.e. $\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \boldsymbol{s}_k$. For this, the right-hand-side of Eq. (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.

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

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

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

GMDD
8, 4401–4451, 2015

5.3 Convergence

Assuming there exist a unique solution of Eq. (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

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

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 $I = 2, \ldots, n_I$ as

$$\varepsilon_I = \|\boldsymbol{y}_I - \boldsymbol{y}_{I-1}\|_2\,.$$

GMDD
8, 4401–4451, 2015

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_I 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 Sect. 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. $\|F(\boldsymbol{y}_k)\|_2$. 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. $\|F'(\boldsymbol{y}_k)\,\boldsymbol{s}_{k,l} + F(\boldsymbol{y}_k)\|_2$.

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

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

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

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

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

GMDD
8, 4401–4451, 2015

6 Software implementation

The toolkit is divided into four repositories, namely `metos3d`, `model`, `data` and `simpack`. The first comprises the installation scripts, the second the biogeochemical model source codes and the third all the data preparation scripts as well as the data. The latter 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. Sect. 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. Sect. 6.4) and to compute interpolation factors and indices (cf. Sect. 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. Sect. 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. Sect. 6.4). The *trans-*

GMDD
8, 4401–4451, 2015

port layer is responsible for reading in the transport matrices, their interpolation to the current time step and their application to the tracer vectors (cf. Sect. 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 Sect. 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.

GMDD
8, 4401–4451, 2015

Regarding the transport, we have $(\mathbf{A}_{\mathrm{imp},j})_{j=1}^{n_{\mathrm{imp}}}$ and $(\mathbf{A}_{\mathrm{exp},j})_{j=1}^{n_{\mathrm{exp}}}$, 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 Sect. 6.5.

Concerning the boundary and domain forcing, we denote the data files by $((\boldsymbol{b}_{i,j})_{j=1}^{n_{\mathrm{b},i}})_{i=1}^{n_{\mathrm{b}}}$ and $((\boldsymbol{d}_{i,j})_{j=1}^{n_{\mathrm{d},i}})_{i=1}^{n_{\mathrm{d}}}$. Here, n_{b} is the number of distinct boundary data sets and $n_{\mathrm{b},i}$ is the number of data files provided for the ith set. Accordingly, n_{d} denotes the number of domain data sets and $n_{\mathrm{d},i}$ 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 $(\boldsymbol{b}_{i,j})_{j=1}^{n_{\mathrm{b},i}}$ we compute the appropriate weights α, β as well as indices j_α, j_β and form the linear combination as

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

GMDD
8, 4401–4451, 2015

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

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

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

$$\left[(\boldsymbol{y}_{1,k})_{k=1}^{n_\mathrm{p}} \cdots (\boldsymbol{y}_{n,k})_{k=1}^{n_\mathrm{p}}\right] \longleftrightarrow \left((\boldsymbol{y}_{i,k})_{i=1}^{n}\right)_{k=1}^{n_\mathrm{p}},$$

where $\boldsymbol{y}_{i,k}$ denotes the kth profile of the ith tracer. Moreover, after the evaluation of the biogeochemical model we reverse the alignment for the transport step. The same situation occurs regarding the domain data. Again, we group the domain data profiles by their profile index k, i.e.

$$\left[(\boldsymbol{d}_{1,k})_{k=1}^{n_\mathrm{p}} \cdots (\boldsymbol{d}_{n_\mathrm{d},k})_{k=1}^{n_\mathrm{p}}\right] \longrightarrow \left((\boldsymbol{d}_{i,k})_{i=1}^{n_\mathrm{d}}\right)_{k=1}^{n_\mathrm{p}}$$

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

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

$$\left[(b_{1,k})_{k=1}^{n_\mathrm{p}} \cdots (b_{n_\mathrm{b},k})_{k=1}^{n_\mathrm{p}}\right] \longrightarrow \left((b_{i,k})_{i=1}^{n_\mathrm{b}}\right)_{k=1}^{n_\mathrm{p}}$$

GMDD
8, 4401–4451, 2015

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 $\boldsymbol{y}_{\text{in}}$.

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 Sect. 7.3 we present results from profiling experiments that show detailed information about the time usage of each operation.

7 Results

In this section, we present results from numerical experiments to verify the software. At first, we use the introduced interface to couple the transport matrix driver with a well

GMDD
8, 4401–4451, 2015

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 verification.

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 (PO_4), dissolved organic phosphorous (DOP) and oxygen (O_2). In fact, we will use just PO_4 and DOP here since the concentrations of DIC, ALK and O_2 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

GMDD
8, 4401–4451, 2015

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. Fig. 1). Note that the Arctic has been filled in. 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 resolu-

GMDD
8, 4401–4451, 2015

tion is at $\Delta t = 1/2880$, which is equivalent to an (ocean) time step of 3 h 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 $\boldsymbol{y}_{PO_4}$ and accordingly $\boldsymbol{y}_{DOP}$, i.e. $\boldsymbol{y} = (\boldsymbol{y}_{PO_4}, \boldsymbol{y}_{DOP})$. 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 $\boldsymbol{y}_{0,PO_4} = 2.17\,\mathrm{mmol\,P\,m^{-3}}$ and $\boldsymbol{y}_{0,DOP} = 0.0001\,\mathrm{mmol\,P\,m^{-3}}$, respectively.

7.2 Solver

We begin our verification by computing a reference solution for the parameter set $\boldsymbol{u}_d$ 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

GMDD
8, 4401–4451, 2015

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 Sect. 7.5.

Nevertheless, the results resemble the observational data taken from the World Ocean Database (Boyer et al., 2013), which were mapped onto a 2.8125° grid and interpolated in space and time for comparison. Figure 3 shows the concentration of phosphate within the first layer. Here, the data is shifted to show Greenwich (0°) at the center. Moreover, Fig. 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 Fig. 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 %.

GMDD
8, 4401–4451, 2015

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 $e_i = 100 \cdot s_i/i$.

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.

GMDD
8, 4401–4451, 2015

$i = 1, \ldots, 260$, we compute the load distribution using Algorithm 2 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 $e_i = 100 \cdot s_i / i$.

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 architecture. 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 Sect. 8. However, since the results give us a good orientation anyway this effect is not investigated further. Overall, as already indicated by the sequential runs, the Intel hardware is the obvious choice for subsequent experiments.

7.5 Convergence control

After a basic verification and a review of technical aspects of our implementation, we investigate the settings to control the convergence of the Newton–Krylov solver. Our intention is to eliminate the over-solving that we observe during the first 200 iterations in Fig. 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 Eq. (6).

GMDD
8, 4401–4451, 2015

Hence, we compute the reference solution from Sect. 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 verification, we are confident that the Newton–Krylov solver is working correctly and, with optimal settings, at least 6 times faster than the spin-up (fixed point iteration).

However, until now we solved the given model equations for the reference parameter set $\boldsymbol{u}_{\mathrm{d}}$ 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

GMDD
8, 4401–4451, 2015
Metos3D
J. Piwonski and T. Slawig

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 verification of the spin-up and the Newton approach, we perform a twin experiment with each solver. We separately compute a reference solution with specific settings and start an optimization run (using the same settings) against it. 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 Sect. 7.5.

We consider the following optimization problem:

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

where

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

and the admissible set is defined as

$$U = \{ \boldsymbol{u} \in \mathbb{R}^m : \boldsymbol{b}_\mathrm{l} \leq \boldsymbol{u} \leq \boldsymbol{b}_\mathrm{u} \}.$$

Here, $\boldsymbol{y}_d = \boldsymbol{y}(\boldsymbol{u}_d)$ is the reference solution computed before and $\boldsymbol{b}_\mathrm{l}$ respectively $\boldsymbol{b}_\mathrm{u}$ 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 $\boldsymbol{u}_0$ (cf. Table 2).

GMDD
8, 4401–4451, 2015

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 result of the optimization run(s) using the spin-up solver. Due to the time limitation of the used batch system, we had to restart the first run, which has been stopped after 200 h. Thereby, the approximation information about the Hessian was lost, which explains the different path that is taken by the second run. However, we observe a convergence of the parameters towards their reference values. At the last optimization step they are $\boldsymbol{u}$ = (0.499, 2.016, 0.670, 0.502, 30.461, 0.019, 0.858). 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. Sect. 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.

GMDD
8, 4401–4451, 2015

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

We primary focused on the technical aspects of the software, the employed solvers and, finally, the usage of each solver for parameter identification. 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 951 000 model years were simulated during the spin-up twin experiment. The approach

GMDD
8, 4401–4451, 2015

may be easy to realize, but it clearly consumes to many computational resources. At least, it shows that the model parameters can be recognized.

Here, the obvious idea would be to take coarser time steps as implied by (Khatiwala, 2007). However, new transport matrices need to constructed for this purpose. Indeed, the appropriate scripts are provided in the `data` repository of Metos3D, but once again, not to further complicate a basic verification the approach was not discussed here.

Moreover, due to the fact that a coarser time step may 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 pre-conditioner implementations or at least the possibility to interact with the inner solver at the appropriate location. However, none of the included PETSc preconditioner nor the presented approach by Khatiwala (2008) is matrix-free. Thus, once again, in order to not further complicate the basic verification 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.

GMDD
8, 4401–4451, 2015

9 Code availability

Name of software: Metos3D (Simulation Package v0.2)
Developer: Jaroslaw Piwonski
Year first available: 2012
Software required: PETSc 3.3
Program language: C, C++, Fortran
Size of installation: 1.6 GB
Availability and Cost: free software, GPLv3
Software homepage: https://metos3d.github.com/metos3d

The toolkit is maintained using the distributed revision control system `git`. All source codes are available at GitHub (https://github.com). The current version has been tagged as `v0.2`. All experiments presented in this work were carried out using this version. The associated material is stored in the `verification` repository.

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

Acknowledgements. The authors would like to thank S. Khatiwala for providing support on the transport matrices and for providing the whole TMM material freely on the internet. Furthermore, both authors would like to thank I. Kriest and A. Oschlies for many fruitful discussions. In particular, Jaroslaw Piwonski would like to thank I. Kriest for teaching him patiently so much about biogeochemical models. At last, we thank our colleague Joscha Reimer for preparing the World Ocean Database data. This work was partly funded by The Future Ocean cluster.

GMDD
8, 4401–4451, 2015

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, Basel, 163–202, 1997. 4404

Balay, S., Brown, J., Buschelman, K., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Smith, B. F., and Zhang, H.: PETSc Users Manual, Tech. Rep. ANL-95/11 – Revision 3.3, Argonne National Laboratory, Lemont, 2012a. 4411

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: 12 July 2013), 2012b. 4404

Bernsen, E., Dijkstra, H. A., and Wubs, F. W.: A method to reduce the spin-up time of ocean models, Ocean Model., 20, 380–392, doi:10.1016/j.ocemod.2007.10.008, 2008. 4403, 4404, 4410, 4421

Boyer, T., Antonov, J., Baranova, O., Coleman, C., Garcia, H., Grodsky, A., Johnson, D., Locarnini, R., Mishonov, A., O'Brien, T., Paver, C., Reagan, J., Seidov, D., Smolyar, I., and Zweng, M.: World Ocean Database 2013, Tech. rep., NOAA Atlas NESDIS 72, s. Levitus, edited by: Mishonov, A., Technical Ed., Silver Spring, 2013. 4422

Bryan, K.: Accelerating the convergence to equilibrium of ocean-climate models, J. Phys. Oceanogr., 14, 666–673, doi:10.1175/1520-0485(1984)014<0666:ATCTEO>2.0.CO;2, 1984. 4403

Danabasoglu, G., McWilliams, J. C., and Large, W. G.: Approach to equilibrium in accelerated global oceanic models, J. Climate, 9, 1092–1110, doi:10.1175/1520-0442(1996)009<1092:ATEIAG>2.0.CO;2, 1996. 4403

Dennis, J. and Schnabel, R.: Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Society for Industrial and Applied Mathematics, Philadelphia, 1996. 4427

Dutkiewicz, S., Sokolov, A. P., Scott, J., and Stone, P. H.: A three-dimensional ocean-seaice-carbon cycle model and its coupling to a two-dimensional atmospheric model: uses in climate change studies, Tech. Rep. 122, MIT Joint Program on the Science and Policy of Global Change, Cambridge, 2005. 4419, 4436

Eisenstat, S. C. and Walker, H. F.: Choosing the forcing terms in an inexact newton method, SIAM J. Sci. Comput., 17, 16–32, doi:10.1137/0917003, 1996. 4413, 4424

GMDD
8, 4401–4451, 2015

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 No. 23/97, JGOFS Bergen, Norway, 1997. 4408
Evans, L.: Partial Differential Equations, American Math. Society, Providence, Rhode Island, USA, 1998. 4412
Fasham, M. J. R. (Ed.): Ocean Biogeochemistry. The Role of the Ocean Carbon Cycle in Global Change., Global Change – The IGBP Series, Springer, Berlin, Germany, 2003. 4403
Fennel, K., Losch, M., Schröter, J., and Wenzel, M.: Testing a marine ecosystem model: sensitivity analysis and parameter optimization, J. Marine Syst., 28, 45–63, doi:10.1016/S0924-7963(00)00083-X, 2001. 4403
Kelley, C. T.: Solving Nonlinear Equations with Newton's Method, SIAM, Philadelphia, USA, 2003. 4404, 4410, 4428
Khatiwala, S.: A computational framework for simulation of biogeochemical tracers in the ocean, Global Biogeochem. Cy., 21, GB3001, doi:10.1029/2007GB002923, 2007. 4403, 4429
Khatiwala, S.: Fast spin up of Ocean biogeochemical models using matrix-free Newton–Krylov, Ocean Model., 23, 121–129, doi:10.1016/j.ocemod.2008.05.002, 2008. 4429
Khatiwala, S.: Transport Matrix Method Web page, available at: http://www.ldeo.columbia.edu/%7Espk/Research/TMM/ (last access: 12 July 2013), 2013. 4403, 4420
Khatiwala, S., Visbeck, M., and Cane, M.: Accelerated simulation of passive tracers in ocean circulation models, Ocean Model., 9, 51–69, 2005. 4403, 4406, 4407
Knoll, D. and Keyes, D.: Jacobian-free Newton–Krylov methods: a survey of approaches and applications, J. Comput. Phys., 193, 357–397, 2004. 4404
Kriest, I., Khatiwala, S., and Oschlies, A.: Towards an assessment of simple global marine biogeochemical models of different complexity, P. Oceanogr., 86, 337–360, doi:10.1016/j.pocean.2010.05.002, 2010. 4403, 4409, 4419, 4428
Kriest, I., Oschlies, A., and Khatiwala, S.: Sensitivity analysis of simple global marine biogeochemical models, Global Biogeochem. Cy., 26, GB2029, doi:10.1029/2011GB004072, 2012. 4419
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. Cy., 20, GB4009, doi:10.1029/2005GB002631, 2006.

GMDD
8, 4401–4451, 2015

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. 4419

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. 4425

Merlis, T. M. and Khatiwala, S.: Fast dynamical spin-up of ocean general circulation models using Newton–Krylov methods, Ocean Model., 21, 97–105, 2008. 4404

Nocedal, J. and Wright, S. J.: Numerical Optimization, Springer, New York, USA, 2000. 4427

Paltridge, G. W. and Platt, C. M. R.: Radiative Processes in Meteorology and Climatology, Elsevier, New York, USA, 1976. 4421

Plato, R.: Concise Numerical Mathematics, 57, American Mathematical Soc., New York, 2003. 4410

Rios, L. M. and Sahinidis, N. V.: Derivative-free optimization: a review of algorithms and comparison of software implementations, J. Global Optim., 56, 1247–1293, 2013. 4428

Saad, Y. and Schultz, M.: GMRES: A Generalized Minimal Residual Algorithm for solving non-symmetric linear systems, SIAM J. Sci. Stat. Comp., 7, 856–869, doi:10.1137/0907058, 1986. 4411

Sarmiento, J. L. and Gruber, N.: Ocean Biogeochemical Dynamics, Princeton University Press, Princeton, USA, 2006. 4403

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, J. Mar. Res., 61, 765–793, 2003. 4403

Siewertsen, E., Piwonski, J., and Slawig, T.: Porting marine ecosystem model spin-up using transport matrices to GPUs, Geosci. Model Dev., 6, 17–28, doi:10.5194/gmd-6-17-2013, 2013. 4418, 4421, 4423

Temam, R.: Navier–Stokes Equations, North-Holland, Amsterdam, the Netherlands, 1979. 4406

Walker, D. W. and Dongarra, J. J.: MPI: a standard Message Passing Interface, Supercomputer, 12, 56–68, 1996. 4404

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. 4403

GMDD
8, 4401–4451, 2015

Zumbusch, G. W.: Dynamic load balancing in a lightweight adaptive parallel multigrid PDE solver, in: PPSC, SIAM, Philadelphia, 1999. 4415

GMDD
8, 4401–4451, 2015

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

GMDD
8, 4401–4451, 2015

Table 2. Parameters implemented in the MITgcm-PO4-DOP model. Specified are the location within the parameter vector, the symbol used by Dutkiewicz et al. (2005), the description of the parameter and the value used for the computation of the reference solution ($\boldsymbol{u}_d$). Shown are furthermore the lower ($\boldsymbol{b}_\mathrm{l}$) and upper ($\boldsymbol{b}_\mathrm{u}$) boundaries as well as the initial parameter guess ($\boldsymbol{u}_0$) used during the twin experiment.

$\boldsymbol{u}$	Symbol	Description	$\boldsymbol{u}_\mathrm{d}$	$\boldsymbol{b}_\mathrm{l}$	$\boldsymbol{u}_0$	$\boldsymbol{b}_\mathrm{u}$	Unit
u_1	κ_{remin}	DOP remineralization rate	0.5	0.25	0.3	0.75	y^{-1}
u_2	α	maximum community production	2.0	1.5	5.0	200.0	$\mathrm{mmol\,P\,m}^{-3}\,\mathrm{y}^{-1}$
u_3	f_{DOP}	fraction of DOP	0.67	0.05	0.4	0.95	1
u_4	κ_{PO_4}	PO_4 half saturation	0.5	0.25	0.8	1.5	$\mathrm{mmol\,P\,m}^{-3}$
u_5	κ_I	light half saturation	30.0	10.0	25.0	50.0	$\mathrm{W\,m}^{-2}$
u_6	k	light attenuation	0.02	0.01	0.04	0.05	m^{-1}
u_7	a_{remin}	power law remineralization coefficient	0.858	0.7	0.78	1.5	1

GMDD
8, 4401–4451, 2015

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**
    $i = \mathrm{floor}(((w + 0.5 * n_{y,k})/n_y) * N)$;
    $n_{p,i} = n_{p,i} + 1$;
    $w = w + n_{y,k}$;
end

GMDD
8, 4401–4451, 2015

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

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

GMDD
8, 4401–4451, 2015




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

GMDD
8, 4401–4451, 2015

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

$\mathbf{q} = \texttt{BGCStep}(t_j, \Delta t, \mathbf{y}_{in}, \mathbf{u}, \mathbf{b}, \mathbf{d})$;

$\mathbf{y}_w = \texttt{TransportStep}(t_j, \mathbf{A}_{exp}, \mathbf{y}_{in})$;

$\mathbf{y}_w = \mathbf{y}_w + \mathbf{q}$;

$\mathbf{y}_{out} = \texttt{TransportStep}(t_j, \mathbf{A}_{imp}, \mathbf{y}_w)$;

GMDD
8, 4401–4451, 2015

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
```


GMDD
8, 4401–4451, 2015

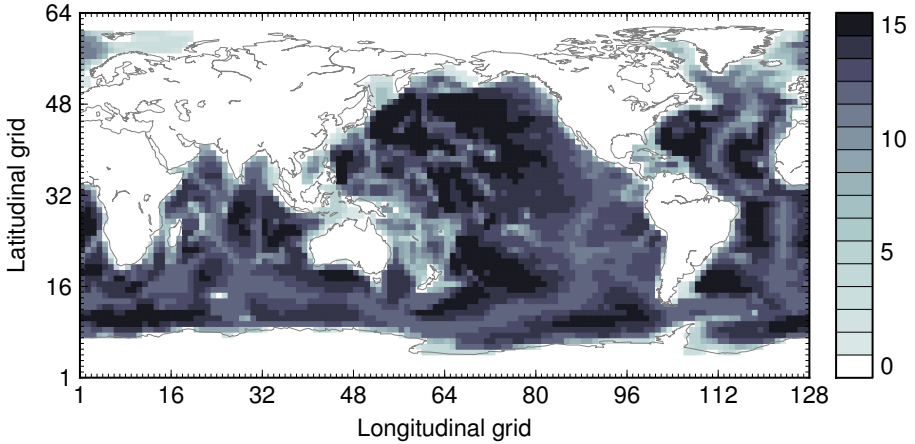

Figure 1. Land-sea mask (geometric data) of the used numerical model. Shown are the number of layers per grip point. Note that the Arctic has been filled in.

GMDD
8, 4401–4451, 2015

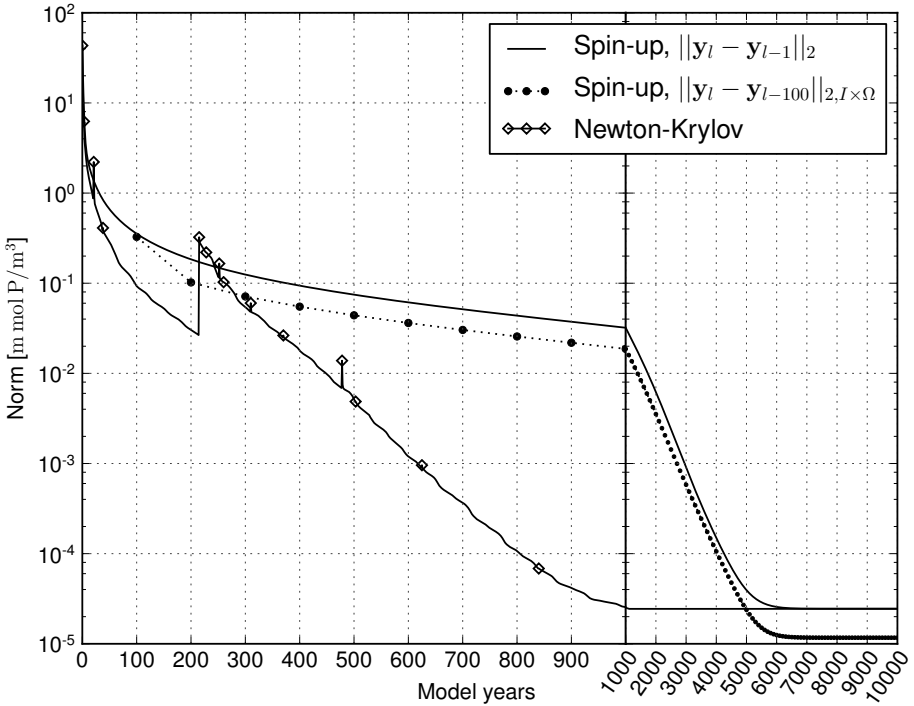

Figure 2. Convergence towards an annual cycle. Spin-up: norm of difference between initial states of consecutive model years (solid line) and trajectories every hundred model years (dots with dashed line). Newton–Krylov: residual norm at a Newton step (diamond) and norm of the GMRES residual during solving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

GMDD
8, 4401–4451, 2015
Metos3D
J. Piwonski and T. Slawig

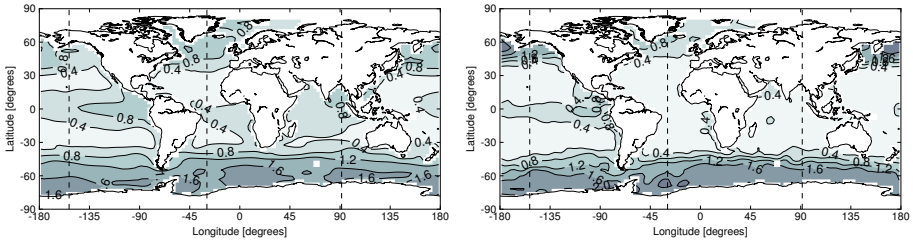

Figure 3. Concentration of phosphate (y_{PO_4}) at the first layer (0–50 m). Left: Shown is the initial state (1 January, 00:00 a.m.) of the converged annual cycle presented in Fig. 2. Right: Interpolated World Ocean Database observational data for the same point in time. The dashed lines depict locations of slices shown in Fig. 4.

GMDD
8, 4401–4451, 2015

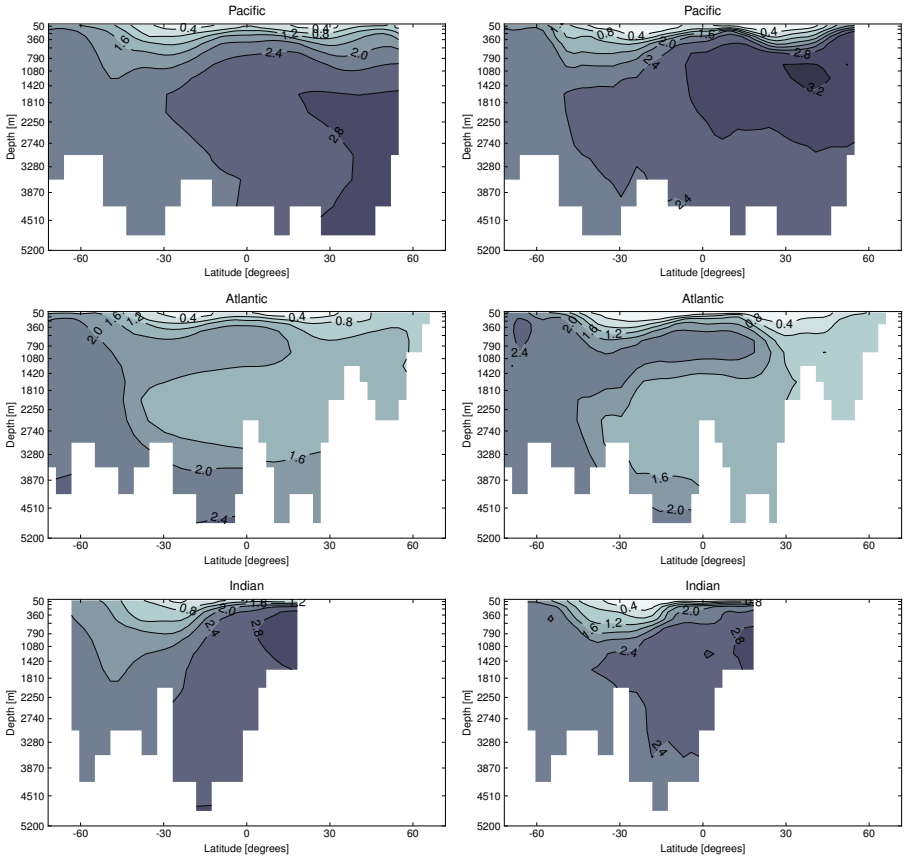

Figure 4. Slices corresponding to Fig. 3: the Pacific (153.2815° W), the Atlantic (29.53125° W) and the Indian (91.40625° E). Left: Simulated tracer concentration. Right: Observational data from the World Ocean Database 2013.

GMDD
8, 4401–4451, 2015

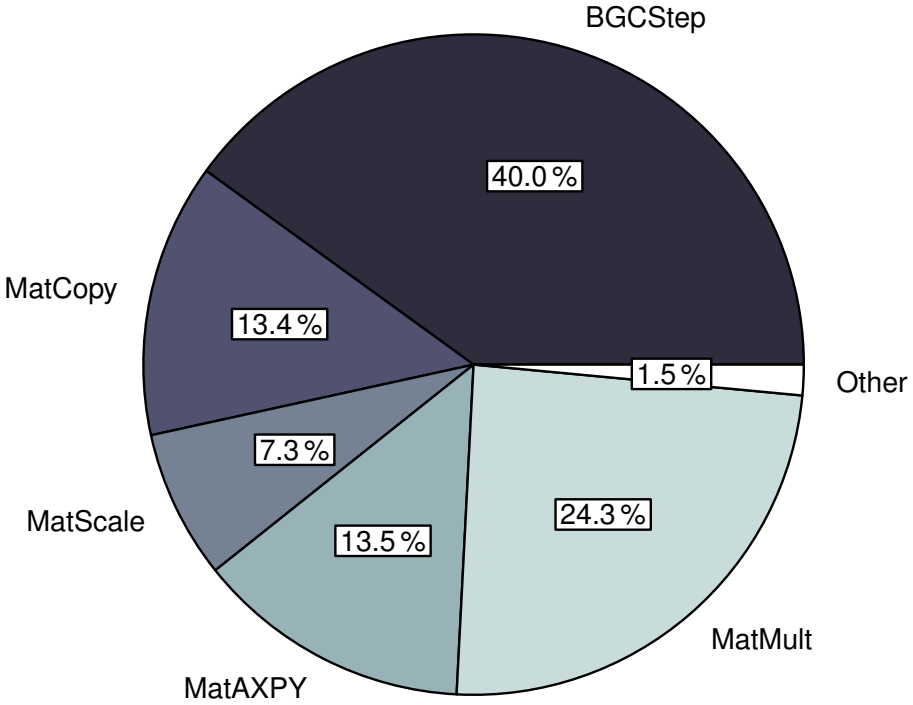

Figure 5. Distribution of the computational time among main operations during the integration of a model year.

GMDD
8, 4401–4451, 2015

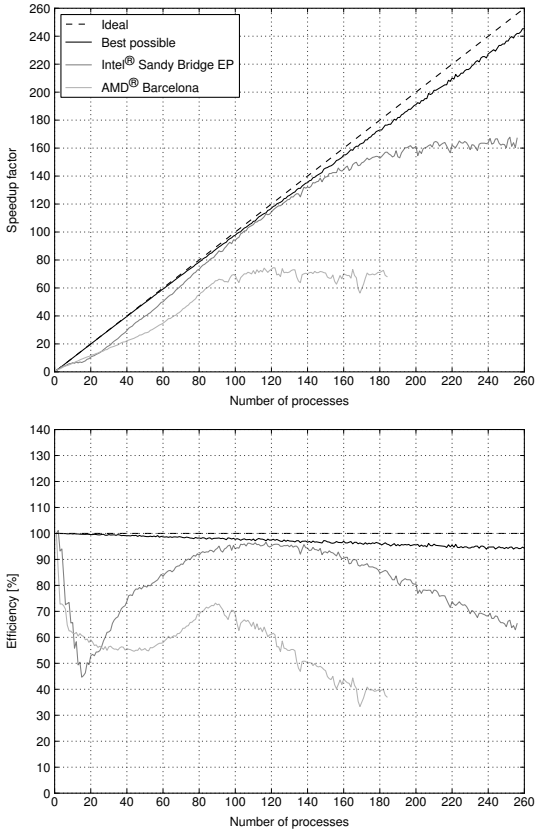

Figure 6. Ideal and actual speedup factor as well as efficiency of parallelized computations. Here, best possible refers to the used load distribution introduced in Sect. 7.4.

GMDD
8, 4401–4451, 2015

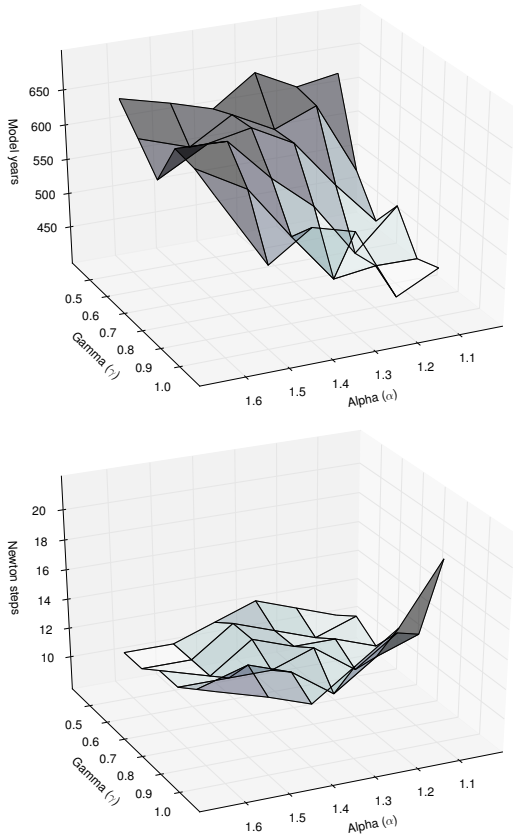

Figure 7. Number of model years and Newton steps required for the computation of the annual cycle $\boldsymbol{y}(\boldsymbol{u}_d)$ as a function of different convergence control parameters α and γ (cf. Eq. 6).

GMDD
8, 4401–4451, 2015

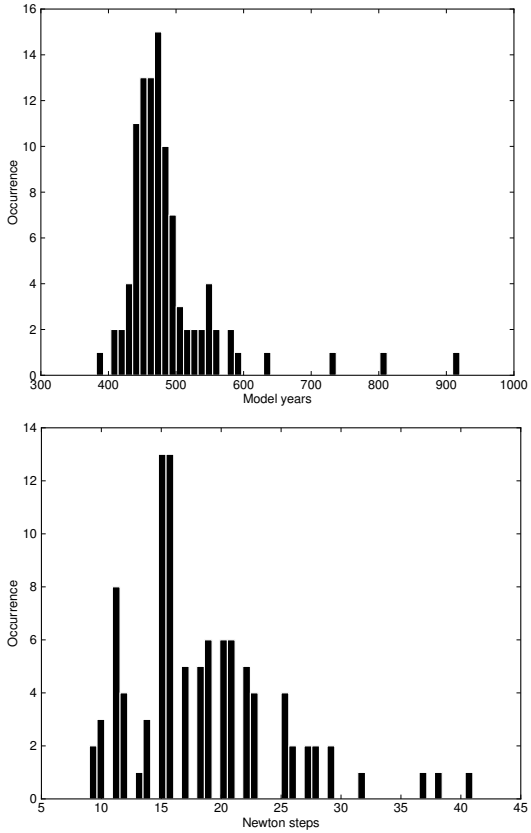

Figure 8. Distribution of number of model years and Newton steps required for the computation of a annual cycle using 100 random parameter samples (cf. Sect. 7.6).

GMDD
8, 4401–4451, 2015

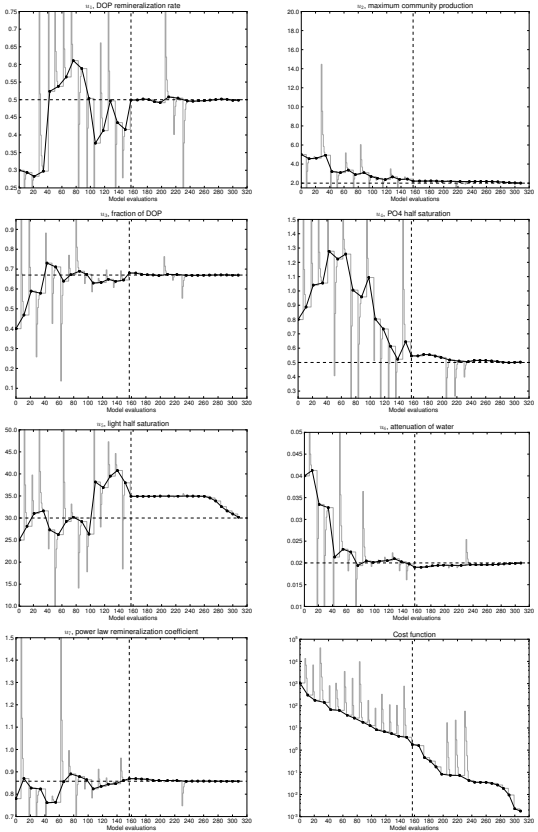

Figure 9. Twin experiment using the spin-up solver. Bullets on the black line depict the steps of the optimization process. The gray line shows all model evaluations including gradient computation and line search step. The dashed line depicts a restart after 157 model evaluations. Vertical limits of the figures are also parameter bounds (except cost function and second parameter).

GMDD
8, 4401–4451, 2015

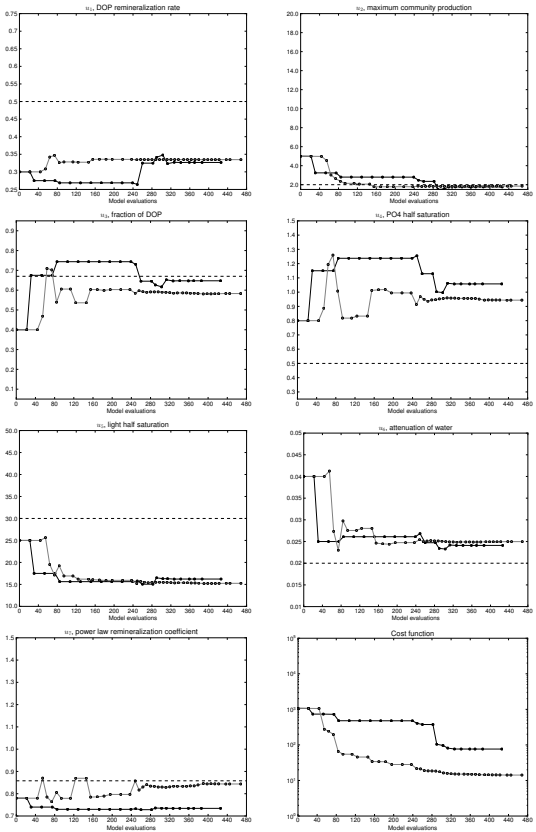

Figure 10. Twin experiment using the Newton solver. The black line depicts the steps taken by the optmizer using a absolute finite difference step that equals to the square root of the machine precision. The gray line refers to a relative finite difference step of 10^{-4}. Intermediate model evaluations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

GMDD
8, 4401–4451, 2015