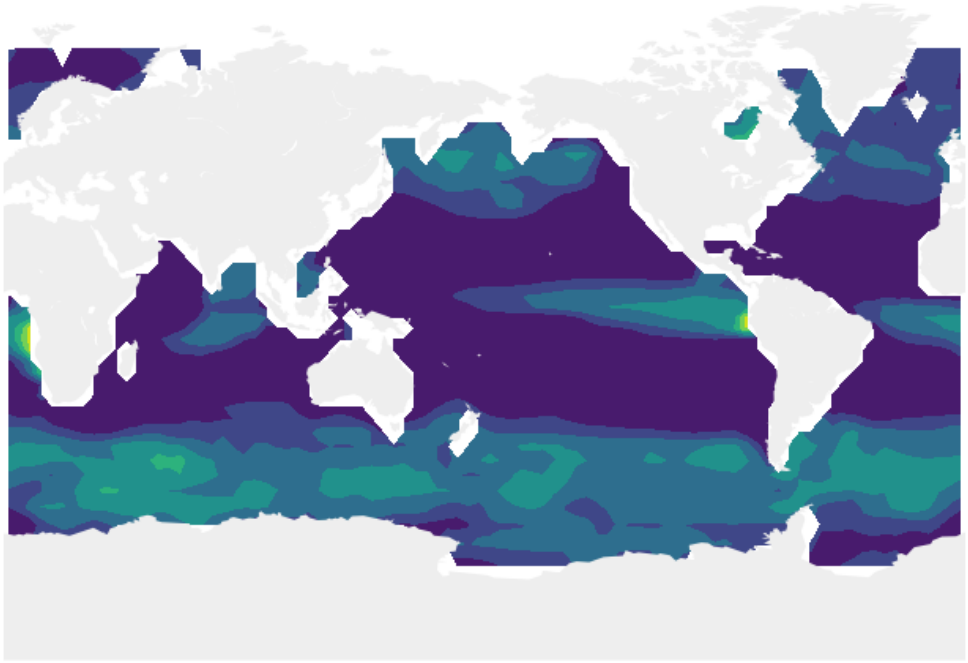
UNIVERSITY OF COPENHAGEN
FACULTY OF SCIENCE



Master Thesis

Steffen Ole Randrup Kristensen - dlx459

Extending the Veros climate simulator with biochemistry

Model design and AMOC collapse

Advisors: Brian Vinter & Markus Jochum

Handed in: September 3. 2019

Contents

1 Introduction **3**

2 The NPZD model **4**
2.1 Sinking of tracers 9

3 Growth limitations **10**
3.1 Zooplankton 10
3.2 Primary production 11

4 Carbon **12**

5 CO_2 exchange between ocean and atmosphere **14**
5.1 Gas exchange 14
5.2 Ocean carbon chemistry 15
5.3 Numerical implementations 21

6 Rule based model **23**

7 Explicit and smoothed representation **29**

8 Object Representation **32**
8.1 Evaluation structure 34

9 Design choices **35**
9.1 Flexibility in model extension 39

10 Diagnostics **41**

11 Employing a user kernel **42**

12 Model evaluation **45**
12.1 Results 46

13 AMOC Collapse **55**
13.1 Model setup 56
13.2 Results 56

14 Conclusion **60**

A Figures with extended dynamics **62**

B Simplified overview of evaluation structure **64**

C Abbreviation list **65**

Abstract

Marine biogeochemistry poses a set of nonlinear coupled problems. Biogeochemistry covers a large variety of components, each with varying importance in different regions. This raises an issue for the models of providing functionality for many possible combinations of actively tracked components and the dynamical systems between them with as little overhead as possible. When models provide multiple configuration options, they must maintain each combination of possible options, which in many models has lead to complex, difficult to maintain code structures. This limits the availability of the model for new users and may require significant introduction times. I propose a different model design, which is able to accommodate configuration options without the combinatorial problems of other designs, and allows for easy addition of new features without the risk of breaking existing functionality. The design is implemented as a module to the "Versatile Ocean Simulation in Pure Python", Veros. I discuss the design choices and requirements for a modern, approachable biogeochemistry simulation model, and present results produced by the implemented design. Finally I simulate a weakening of the Atlantic meridional overturning circulation and analyze the results on global biogeochemistry.

1 Introduction

Veros is a modern general circulation model, which aims to be easy to use by both students and researchers while supporting a multitude of configurations for realistic or idealized setups (Häfner, Jacobsen, Nuterman, et al., 2018). This project extends the functionality of Veros to additionally support simulations of biogeochemistry, which is a requirement for climate research as well as modelling of natural systems. The successful implementation follows the vision of Veros by being easy to use, verify and modify.

Biogeochemistry forms part of several global circulation systems. Plankton species are responsible for production and consumption of nutrients like phosphate and nitrate as well as carbon species which are relevant in respect to absorption and out gassing of carbon dioxide and is of importance to climate research. Biogeochemistry is also important for research in ecosystems, and oil prospecting, and as an interdisciplinary discipline useful in several fields including but not limited to atmospheric sciences, biology, and oceanography. Models working within those fields may thus be expected to provide capabilities in the area of biogeochemistry. A successful model is able to accommodate contributions by contributers from multiple different backgrounds with differing requirements and easily allow users to create and use simulations. The model should facilitate contributions and use by experts as well as students. Biogeochemistry is implemented in several current global simulation models like University of Victoria Earth System Climate Model (*UVic ESCM* 2018). These models have grown over several years, allowing for increasingly complex model configurations while maintaining speed and correctness. As the models have grown and functionality has been added, it has become necessary to support several different configuration options. This has in some cases made the code base hard to maintain and extend, because any extension must be added to each of the possible combinations of configuration options. Frequently this results in the same variable being updated by nearly identical equations in multiple locations within the code. Such examples makes the model hard to maintain and reduces the accessibility to new users and contributers. The solution proposed in this project and implementation in Veros is a complete modularization, which describes every interaction between tracers as a rule and the full dynamics of the system as a set of rules.

In order to adhere to Veros' focus on simplicity, usability and adaptability, the biogeochemistry module facilitates constructing complex model setups with multiple tracers with dynamic interactions tailored for a specific environment without disrupting the simplicity of more basic setups.

While the dynamics of the biogeochemistry module follow the mathematical description of the implementation in UVic ESCM(A. Schmittner, A. Oschlies, et al., 2005; Andreas Schmittner, 2018), the programmatic implementation has been redesigned to allow for extensibility without reducing readability and at the same time provide simple configuration for users.

This project describes the implementation of the biogeochemistry module in Veros and how it separates tracer interactions from the logical model structure in order to keep readability and maintainability as well as allowing for easy extension of the model or selecting the configuration required for a desired simulation. I discus considerations in designing the model in terms of manageability and performance. The model design is intended to be easy to use for students or contributers with limited programming experience while being powerful and extensible for users with high demands.

The model is evaluated in its correctness by comparing the output of a 400 year simulation to known datasets, and performing a simulation, which weakens the Atlantic Meridional Overturning Circulation.

All code used to produce the biogeochemistry module described in this document is available at https://github.com/SteffenRandrup/Veros/tree/mobi

2 The NPZD model

The Nutrients-Phytoplankton-Zooplankton-Detritus model, from here on referred to as the NPZD model, describes the growth patterns of phytoplankton and zooplankton given nutrients and the buildup of detritus, dead organic matter, produced from the dying plankton, which in turn is remineralized into nutrients. Figure 1 presents a graphical representation of the internal dynamics of the biogeochemical system.

The underlying mathematical description of ocean biogeochemistry for the NPZD model implemented in Veros follows that of UVic ESCM (Andreas Schmittner, 2018). In this section I introduce the model, its components, and internal dynamics. Concentrations of each of the components included in the NPZD model are tracked in every cell in the Arakawa C-Grid, which Veros uses for its tracer variables. A tracer is the collection of concentrations of a model variable within each cell in the grid. In figure 1 tracers are marked with ellipses. As in UVic ESCM the time evolution of any tracer consists of two parts: Physical transport and biological activity. Transport and the internal biological dynamics may be separated from each other, which allows treating each separately.

$$\frac{\partial C_i}{\partial t} = T_i + S_i \tag{1}$$

C_i is the concentration of tracer i, T_i is turbulent terms, S_i is the change in concentration due to biological activity which in the following may be referred to as source-minus-sink. Equation 1 applies within a single cell and the full grid. The transport term, T_i, uses the general scheme for temperature and salinity transport already present in Veros, which supports advection and isopycnal diffusion as used for this project as well as multiple other diffusion schemes and IDEMIX (Häfner, Jacobsen, Eden, et al., 2018). The bigeochemistry module also supports

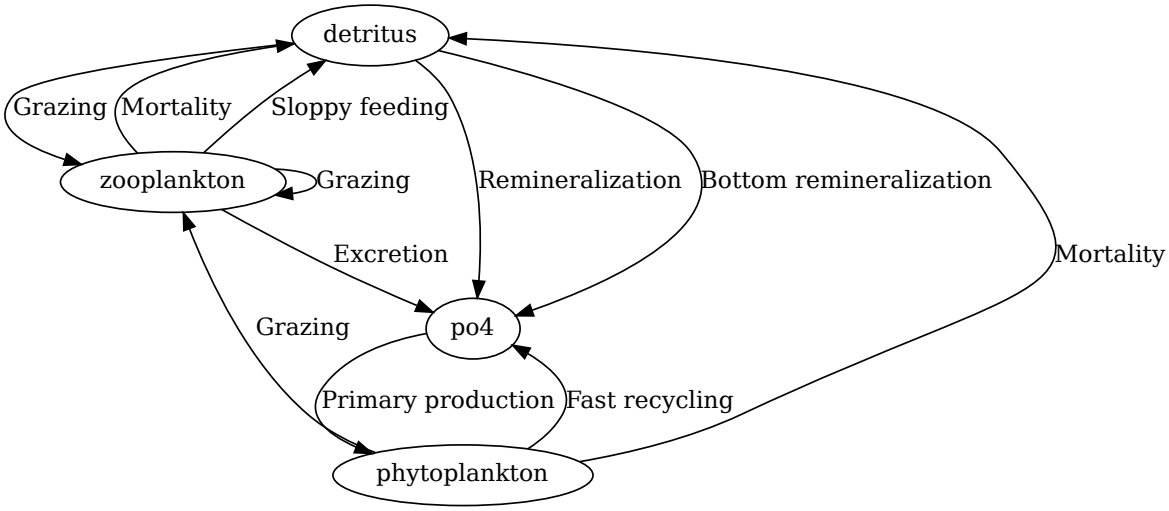


Figure 1: Auto-generated figure of the NPZD dynamics considers mortality, grazing, recycling, primary production and sinking

omitting physical transport for individual tracers, which may be desirable in advanced model configurations as described in section 7.

The mathematical description of the biogeochemistry extension to Veros follows that of UVic ESCM, but the implemented model differs significantly. The design is based on an observation, that the dynamics are confined to a closed system, which always carries change in tracers from one tracer to another, represented by the arrows in figure 1. The implementation details of the biogeochemistry module requires creating rules for each interaction between tracers. This concept is explained in detail in section 6. In this section, I will describe the biogeochemical system by the changes to each tracer as the conversion of one tracer to another. In doing so, I will provide Python functions illustrating the modelled behaviour. The functions form the basis of the mentioned rules. They return Python dictionaries who's keys are the names of the tracers being changed. The value belonging to that key is the time derivate of the tracer concentration. By including the functions, I will show, that it is possible to describe every interaction as a flow from one tracer to another. In section 6 I will argue, why this approach is better than other current model designs. The presented functions are included as is from the current code base. They include references to variables, which are calculated elsewhere to avoid duplicate computations and allowing them to be available for diagnostics. The calculations involved in getting these variables are as presented in the surrounding description.

Each of the tracers in the model interact with each other, one tracer acts as the source of growth in the other, the sink. Most of the dynamics happen within a single grid cell and is thus independent of the concentrations of tracers in other cells. Exceptions to this are: Material sinking to the bottom will gain material from the cell above and lose material to the cell below. This doesn't influence the dynamics directly, but provides an additional contribution to the development of the tracer. The other exception is primary production. Primary production is the growth of phytoplankton by photosynthesis and nutrient consumption. Because biological tracers and water attain part of the incoming photosynthetically active radiation, the primary

production in a cell is influenced by the concentration of light attenuating tracers in cells above it. (Andreas Oschlies and Garçon, 1999; A. Schmittner, A. Oschlies, et al., 2005; Andreas Schmittner, 2018)

In the following the source-minus-sink of the tracers in the NPZD model will be described. I will build the full interaction set gradually by describing the interactions of a single tracer one at a time. I will denote the tracers as follows: Phosphate is PO_4, detritus is denoted D, P is for phytoplankton and zooplankton is Z. Grazing by zooplankton on other tracers behaves identically for every tracer and will be presented as $G(X)$, where X is the tracer being grazed upon. The grazing behaviour is described in section 3. $R_{Y:Z}$ is the Redfield ratio of Y to Z. The Redfield ratio is a presumed fixed average ratio between presence of elements in the ocean (Martiny, Vrugt, and Lomas, 2014). While describing the effects of the rules, I will provide code samples for adding the functionality to the biogeochemistry model in Veros. I will expand upon the reasoning behind the model design in section 6.

Starting with phosphate, concentration can increase from remineralization of detritus given by a temperature dependant remineralization rate multiplied by the concentration of detritus, $\mu_D D$. Remineralization is the effect of a microbial loop which consumes dead organic matter and produces nutrients (A. Schmittner, A. Oschlies, et al., 2005).

The model describes this process as a conversion of detritus to phosphate. As the process of remineralization is common among several recycling processes, a general recycling function has been created, which is shown in listing 1.

Listing 1: General recycling

```
@veros_method(inline=True)
def recycling(vs, plankton, nutrient, ratio):
    """Plankton or detritus is recycled into nutrients"""
    return {nutrient: ratio * vs.recycled[plankton], plankton: - vs.recycled[plankton]}
```

The general recycling function reuses a variable storing the amount of recycled material for the tracer in question. This amount has been pre-calculated, as it is useful in other nutrient recycling calculations. For detritus it is calculated as $\mu_{D0} \cdot b^{cT} D$. With T the temperature in the cell and b and c being model settings modifying the temperature dependence. In section 8 I will further explain the recycling methods for different tracers and how I have designed the biogeochemistry module to allow easy modification for other tracers. The function for the rule describing the remineralization process can then be created like in listing 2.

Listing 2: Remineralization of detritus to PO_4

```
@veros_method(inline=True)
def recycling_to_po4(vs, plankton, phosphate):
    """Recycling to phosphate is scaled by redfield ratio P to N"""
    return recycling(vs, plankton, phosphate, vs.redfield_ratio_PN)
```

When a function like listing 2 has been created, it must be registered in Veros as a rule, as done in listing 4, and activated to work in the dynamical system. I will present the concept of creating, registering, and activating rules in section 6. In the following it is assumed that every function presented is registered as a rule and activated.

Zooplankton produces phosphate by excreting a fraction of what it consumes by cellular respiration and waste dumping which is then modelled as being instantly available as nutrients (A. Schmittner, Gruber, et al., 2013).

Listing 3: Zooplankton excretion

```
@veros_method(inline=True)
def excretion(vs, zooplankton, nutrient):
    """Zooplankton excretes nutrients after eating. Fecal matter, breathing..."""
    return {zooplankton: - vs.excretion_total, nutrient: vs.redfield_ratio_PN * vs.
        excretion_total}
```

Fast remineralization of phytoplankton via microbial effects contributes to phosphate growth by remineralizing part of the phytoplankton population at a rate of $\mu_{Pt}P$ (Kvale et al., 2015). The rule to describe fast recycling of phytoplankton builds on the same recycling function as remineralization of detritus, listing 2, only the rule describing the interaction specifies phytoplankton as the source rather than detritus. Therefore the same function can be used when registering two different rules which specify each their source and sink.

Listing 4: The same function can be used in multiple rules

```
register_npzd_rule(vs, 'remineralization', (recycling_to_po4, 'detritus',
'po4')) # register remineralization rule for detritus -> po4
register_npzd_rule(vs, 'fastrecycling', (recycling_to_po4, 'phytoplankton',
'po4')) # rule for fast recycling of phytoplankton to po4
```

Phosphate concentration decreases from production of phytoplankton via photosynthesis at a rate of $J \cdot P$ which will be described further in section 3.2. The function to describe the interaction, once the primary production has been calculated, is similar to that of the other functions, I have created so far.

Listing 5: Primary production

```
@veros_method(inline=True)
def primary_production(vs, nutrient, plankton):
    """Primary production: Growth by consumption of light and nutrients"""
    return {nutrient: - vs.redfield_ratio_PN * vs.net_primary_production[plankton],
        plankton: vs.net_primary_production[plankton]}
```

Combining the contributions to the consumption and production of phosphate creates an equation for the source-minus-sink for phosphate. Example 1 in section 6 describes how the rules set constructs the full source-minus-sink equations.

$$S(\mathrm{PO_4}) = R_{P:N}\left(\mu_D D + \gamma_1(1 - ge_Z)\left(G(P) + G(Z) + G(D)\right) - JP + \mu_{Pt}P\right) \tag{2}$$

The decrease in phosphate by primary production is balanced by an increase in phytoplankton. Because this interaction was already added in listing 5, the corresponding source-minus-sink term for phytoplankton is already in the model.

The phytoplankton population size decrease by mortality due to old age, diseases, fatal

damage etc. of the plankton turning it into detritus at a rate $\mu_P P$. Concentrations are also reduced by grazing by zooplankton. Part of the phytoplankton is immediately recycled to nutrients in fast recycling $\mu_{Pt} P$ as previously mentioned. For a full equation for source minus sink of phytoplankton the remaining two terms must be added. The full equation becomes that of equation (3) while the two terms, which have not yet been turned into function are presented in listings 6 and 7

$$S(P) = JP - \mu_P P - G(P) - \mu_{Pt} P \tag{3}$$

Listing 6: Grazing on phytoplankton

```
@veros_method(inline=True)
def grazing(vs, eaten, zooplankton):
    """Zooplankton grows by amount grazed, eaten decreases by same amount.
    Zooplankton growth is reduced by rules for excretion and sloppy feeding.
    The actual zooplankton growth is available in vs.digested"""
    return {eaten: - vs.grazing[eaten], zooplankton: vs.grazing[eaten]}
```

Listing 7: Phytoplankton mortality

```
@veros_method(inline=True)
def mortality(vs, plankton, detritus):
    """All dead matter from plankton is converted to detritus"""
    return {plankton: - vs.mortality[plankton], detritus: vs.mortality[plankton]}
```

The zooplankton population's only source of growth is grazing. It may graze on phytoplankton, detritus and even itself. Grazing is not entirely efficient. Only a fraction γ_1 of the grazed amounts are ingested. The remaining fraction is lost by sloppy feeding to detritus. Of the ingested material a fraction, ge_Z, induces growth in the zooplankton population and the remaining fraction is converted to nutrients. I described this process in the source-minus-sink description for phosphate. Like phytoplankton, zooplankton may die from old age, disease, and physical damage, but is modelled with a quadratic mortality rate, $\mu_Z Z^2$.

$$S(Z) = \gamma_1 \cdot ge_Z \left(G(P) + G(Z) + G(D)\right) - \mu_Z Z^2 - G(Z) \tag{4}$$

The function for describing zooplankton mortality is implemented exactly like the one for phytoplankton, because mortality calculations are pre-calculated for use in multiple rules.

The rule for zooplankton grazing on itself should be implemented like the other grazing rule, however the increase and decrease in population is balanced out. Therefore zooplankton grazing on other zooplankton causes a net decrease in concentration, which is accounted for in the rule for excretion and for sloppy feeding.

The two plankton concentrations decreased by mortality of the plankton. This is reflected in the detritus calculations. As was described for zooplankton, some of the grazed material is not ingested by the zooplankton. In stead it is converted to detritus by sloppy feeding. In order to complete the description, this final term must be included.

Listing 8: Sloppy feeding

```
@veros_method(inline=True)
def sloppy_feeding(vs, zooplankton, detritus):
    """ When zooplankton graces some amount is not ingested. This is converted
    to detritus """
    sloppy_sum = sum(vs.sloppy_feeding.values())
    return {zooplankton: - sloppy_sum, detritus: sloppy_sum}
```

The recycling of detritus contributes to increase in nutrient concentration and must be matched by a decrease in detritus concentration. The last term of the detritus source-minus-sink represents sinking of the detritus. Unlike the other tracers detritus falls towards the ocean bottom. The sinking speed w_D increases with depth. Sinking is described further in section 2.1. The source-minus-sink equation for detritus is then:

$$S(D) = \mu_P P + (1 - \gamma_1)\left(G(P) + G(Z) + G(D)\right) + \mu_Z Z^2 - \mu_D D - G(D) + w_D \frac{\partial D}{\partial z} \tag{5}$$

The dynamics of the NPZD model described in equations (2) to (5) and the code listings in this section are depicted in figure 1, which has been generated from the rules describing the interactions between tracers as described in this section.

In section section 4 I will extend the model by a carbon cycle. In doing so, the presented equations for the dynamics will be extended by additional terms, but the dynamics and the functions describing it defined in this section do not change. The power of the implemented design lies in allowing for easy extension without reference to implementation details of other parts.

2.1 Sinking of tracers

Most tracers considered in the biogeochemistry module are considered neutrally buoyant. If the transport terms were not applied, every cell would be independent. However, such an assumption is not always valid. Some tracers need to sink towards the ocean bottom. For the purposes of this work it is convenient to only treat detritus as such. Sinking of detritus is described by its sinking speed and the size of the grid cells.

Consider the amount of detritus in a cell as a box with volume $A \cdot dz$. During time dt the distance the box sinks $w_D \cdot dt$, with w_D the sinking speed of detritus. The fraction of the box, which has moved from cell i to cell $i-1$ below is then $\min\left(1, \frac{w_D dt}{dz}\right)$. Since the module tracks concentration of tracers, not absolute value, and has a grid with variable depths of cells, the concentration exported from layer i to layer $i-1$ must be scaled by their respective volumes. The exported concentration from layer i during the time dt is then calculated as

$$\text{export}(C_i) = C_i \frac{w_D}{dz_i} dt \tag{6}$$

Exported material from layer $i+1$ is imported into layer i below.

$$\text{import}(C_i) = \text{export}(C_{i+1}) \frac{A dz_{i+1}}{A dz_i} = \text{export}(C_{i+1}) \frac{dz_{i+1}}{dz_i} \tag{7}$$

At the surface import is set to 0 and at the bottom export is set to 0. The amount, that would have been exported, had there been a layer below, is then remineralized fully into nutrients.

For detritus the sinking speed increases by depth until a maximum depth, mwz, at which point the speed remains constant.

$$w_D = w_{D0} + mw \cdot \min(z_w, mwz) \tag{8}$$

w_{D0} is the surface sinking speed, mw the increase in speed with depth, z_w the depth at the top of the grid box, and mwz the depth below which the sinking speed remains constant.

The derivation has considered detritus explicitly, but is valid for any tracer.

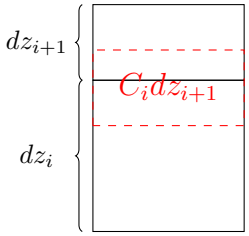


Figure 2: Sinking of tracer with concentration C_i should be scaled by layer depth

3 Growth limitations

The model description in section 2 explained the terms in the source-minus-sink equations in general terms in order to explain the concept of carrying tracer concentrations between a source and sink. In this section I further explain, what is included in each term.

Many of the parameters used in the biogeochemistry module are temperature dependant. For example the remineralization rate of detritus and fast recycling rate of phytoplankton both are. They each present a base rate controlled by the temperature dependent factor b^{cT}. The remineralization rate of detritus is thus calculated as $\mu_D = \mu_{D_0} \cdot b^{cT}$, where b and c are model parameters for tuning the temperature dependence and T is temperature. For the fast recycling rate of phytoplankton, the equation is similar. $\mu_P = \mu_{P_0} \cdot b^{cT}$. This form of temperature dependence is also used in the calculations for primary production and zooplankton grazing.

Whereas the remineralization and fast recycling rates are only controlled by temperature, primary production and grazing are controlled by additional factors.

3.1 Zooplankton

Zooplankton grazing is partly temperature dependent but is also dependent on the availability of prey. A preference for each type of prey is assigned, such that the sum of preferences is 1. Grazing on each type of prey then scales with the concentration of the prey and the preference for grazing on it. equation (9) shows the full equation for calculating the grazing factor $G(X)$ on prey X.

$$G(X) = \mu_Z^{\max} Z X \frac{\psi_X}{\sum_i \psi_i \text{prey}_i + k_Z} \tag{9}$$

ψ_X is the preference for zooplankton to eat prey of type X. The denominator represents all the available prey scaled by the preference factor plus the half-saturation constant. $\mu_Z^{\max}$, the

limiting factor is a temperature dependant measure of the maximum potential grazing rate. The temperature dependence is similar to that of the other temperature dependent terms, although it is capped at a maximum temperature of 20 degrees Celsius.

$$\mu_Z^{\max} = \mu_Z b^{c \min\{20,T\}} \tag{10}$$

Additionally, the zooplankton growth is limited by its ability to assimilate prey and how much of the ingested prey is converted to growth of zooplankton. These limitations were presented in the previous section.

3.2 Primary production

The growth rate for phytoplankton is dependent on three factors: Availability of photosynthetically active radiation, temperature and presence of nutrients. The calculations of the effective growth rate are split into two parts. The first part considers only light limited growth, that is the absence of light is the primary limiting factor in plankton growth. Shortwave radiation intensity is given as input to the biogeochemistry model. For plankton growth below the surface, the downward portion of the incoming radiation is the effective available radiation for primary production. As light reaches the atmosphere-ocean boundary, it is refracted and alters the effective downward component. This is dependent on the angle of incidence and refractive index of the ocean-atmosphere boundary. The refractive index is fixed at $n = 1.33$, whereas the angle of incident light varies throughout the year and by latitude. The declination of the incident light, δ is determined by equation (11).

$$\delta = \sin\left(\left(\frac{t}{1\text{y}} \bmod 1 - \phi\right) \cdot 2\pi\right) \cdot \varepsilon \tag{11}$$

Here t is model time, ϕ shifts the time of year, at which the declination will reach its extremes to the equinoxes on 21 March $\left(\frac{t}{1\text{y}} = 0.22\right)$ and 21 September. The calculations must also consider Earth's obliquity, $\varepsilon = 23^\circ \approx 0.4$, which scales the effective declination. Knowing the declination, the effective vertical coordinate $\hat{z}$ which takes refraction into account may be calculated.

$$\hat{z} = z\sqrt{1 - \left(1 - \cos^2 \max\left(-1.5 \min\left(1.5, y\right) - \delta\right)/n^2\right)} \tag{12}$$

With $n = 1.33$ the index of refraction relating the angle of incidence in air to the angle of incidence in water.

As light passes through the water column a fraction of it is retained by the water. Phytoplankton also blocks light as it absorbs it for primary production. The configurations described in this project do not include any other light attenuating tracers and can be described by equation (13). However the model is designed for easy extension and allows any tracer to attenuate or block light, therefore the general model description is that of equation (14). Section 8 describes how the model adds flexibility of additional tracers to attenuate light.

$$I = I_{z=0} \exp\left(-k_w \hat{z} - k_c \int_0^{\hat{z}} P dz\right) [1 - a_i] \tag{13}$$

a_i is the fractional ice cover. As Veros does not yet have a sea ice model, a_i is set to 1 where the sea surface temperature is less than $-1.8^\circ C$ and temperature forcing is negative. The

amount of available shortwave radiation at depth z for any model setup with tracers T_j and light attenuation factor k_j is then.

$$I = I_{z=0} \exp\left(-k_w \hat{z} - \sum_j k_j \int_0^{\hat{z}} T_j dz\right) [1 - a_i] \tag{14}$$

The effective primary production rate J is calculated in two parts. One where nutrient availability is the primary limiting factor, and one where the absence of photosynthetically active radiation acts as the main limiting factor to growth. Generally any nutrient (N) with half-saturation constant k_N consumed by the plankton provides a limiting factor of $\frac{N}{k_N + N}$. The NPZD model in section 2 only contains one nutrient, PO_4. When nutrient N has the smallest limiting factor of all the nutrients, the effective primary production rate becomes

$$J(I, N) = \min\left(J_I, J_{max} \frac{N}{k_N + N}\right) \tag{15}$$

J_{max} is the light-saturated growth, which depends only on temperature. It takes on the same form as the temperature dependence terms defined for other temperature dependent terms

$$J_{max} = ab^{cT} \tag{16}$$

Where a is the maximum growth rate of the phytoplankton type. The light-limited growth takes the form of

$$J_I = \frac{J_{max} \alpha I}{\left[J_{max}^2 + (\alpha I)^2\right]^{0.5}} \tag{17}$$

Here α is the initial slope of the photosynthesis-irradiance curve.

To complete the calculation of the primary production equation (15) is averaged over depth during a triangular shaped diurnal cycle (A. Schmittner, A. Oschlies, et al., 2005).

$$\begin{aligned} J_I^{avg} &= \frac{1}{\Delta z \cdot 24h} \int_{z-\frac{\Delta z}{2}}^{z+\frac{\Delta z}{2}} \int_0^{24h} J_I dz dt \\ &= \frac{G_D}{k_w \Delta z} \left[\Phi\left(\frac{2G_I}{G_D}\right) - \Phi\left(\frac{2G_I}{G_D} e^{-(k_w + k_c P)\Delta z}\right)\right] \end{aligned} \tag{18}$$

Where Δz is the layer depth. The length of days changes throughout the year, and so considering days with sunlight of fractional time $d = \arccos(-\tan\phi \tan\delta)$ the growth is then $G_D = J_{max} d$. The function $\Phi(u)$ is given by

$$\Phi(u) = \ln\left(u + \sqrt{1 + u^2}\right) - \left(\sqrt{1 + u^2} - 1\right)/u \tag{19}$$

4 Carbon

As described in section section 2 phosphate is produced by remineralization of detritus, fast recycling of phytoplankton and excretion by zooplankton. Phosphate is consumed by growing phytoplankton in primary production.

The same arguments apply to the production and consumption of dissolved inorganic carbon, DIC. In the case of DIC however, a few other processes must also be considered. When plankton is remineralized a portion of the nutrients produced is calcium. Calcium bonds with carbonate to produce calcium carbonate, $CaCO_3$. The production of calcium carbonate reduces the concentration of DIC. An external source of carbon is the solution of carbon dioxide from the atmosphere into the ocean surface layer. The flux of carbon over the ocean-atmosphere boundary is described in detail in section 5. The flux is among other factors dependent on the alkalinity of the ocean. Therefore both alkalinity and DIC are tracked. Production of calcium carbonate removes carbonic acid, a double proton acceptor, causing a reduction in alkalinity twice that of the reduction in DIC. With the setup described until now, it is not possible to make a full description of the calcium carbonate concentrations. However it is possible to describe the effects of the involved tracers on the $CaCO_3$ concentration and how this reduces or increases the concentration of DIC and alkalinity.

$CaCO_3$ is modelled as being produced in the same processes as detritus. A ratio of the plankton lost to mortality is remineralized to calcium, which then produces calcium carbonate using available DIC. Plankton lost to sloppy feeding by zooplankton is likewise remineralized, producing calcite.

The calcite production may then be written as as equation (20) with Ca_{pr} the calcium production ratio

$$Pr(\mathrm{CaCO_3}) = \left(\mu P + \mu_z Z + (1 - ge_Z)\left(G(P) + G(Z)\right)\right) \mathrm{Ca}_{pr} \cdot R_{C:N} \tag{20}$$

The produced calcium carbonate forms calcite and sinks. After formation it is later dissoluted and again added to DIC and alkalintiy. The dissolution happens faster at depth, which is modelled without tracking the concentration of $CaCO_3$ explicitly by redistributing the produced calcite according to equation (21) (A. Schmittner, Gruber, et al., 2013).

$$Di(\mathrm{CaCO_3}) = \int Pr(\mathrm{CaCO_3})dz \cdot \frac{d}{dz}\left(e^{-z/d_{CaCO3}}\right) \tag{21}$$

equation (21) preserves the total produced calcite, but distributes it in larger amounts towards the ocean bottom.

The source-minus-sink of $CaCO_3$ can then be set to

$$S(\mathrm{CaCO_3}) = Pr(\mathrm{CaCO_3}) - Di(\mathrm{CaCO_3}) \tag{22}$$

Which can be included in the source-minus-sink equations for DIC and alkalinity

$$S(\mathrm{DIC})_{base} = R_{C:N}\left(\mu_D D + \gamma_1(1 - ge_Z)\left(G(P) + G(Z) + G(D)\right)Z - JP + \mu_{Pt}P\right) + F_{\mathrm{CO_2}} \tag{23}$$

$$S(\mathrm{DIC}) = S(\mathrm{DIC})_{base} - S(\mathrm{CaCO_3}) \tag{24}$$

$$S(\mathrm{Alk}) = -S(\mathrm{DIC})_{base} - 2S(\mathrm{CaCO_3}) \tag{25}$$

The dissoultion and production of $CaCo_3$ effectively redistributes it. The production removes DIC and alkalinity at locations, where calcite is created as described by equation (20), which would be primarily near the surface layer where phytoplankton and zooplankton are present. This effectively removes DIC and alkalinity at the upper layers.

The produced calcite is dissolved in larger amounts at lower depths as described by equation (21). The two processes described effectively removes DIC from the surface and adds it back at lower depths. When $CaCO_3$ is not tracked explicitly, the $CaCO_3$-tracer only represents the amounts produced, which is then used in calculating the dissolved amounts for every level. The tracer must then be reset after each time step. This process and reasoning behind is described further in section 7.

When all the rules for the described processes have been created, the graph in figure 1 is extended to figure 3. Note that the dynamics described by figure 1 is still present in the updated graph.

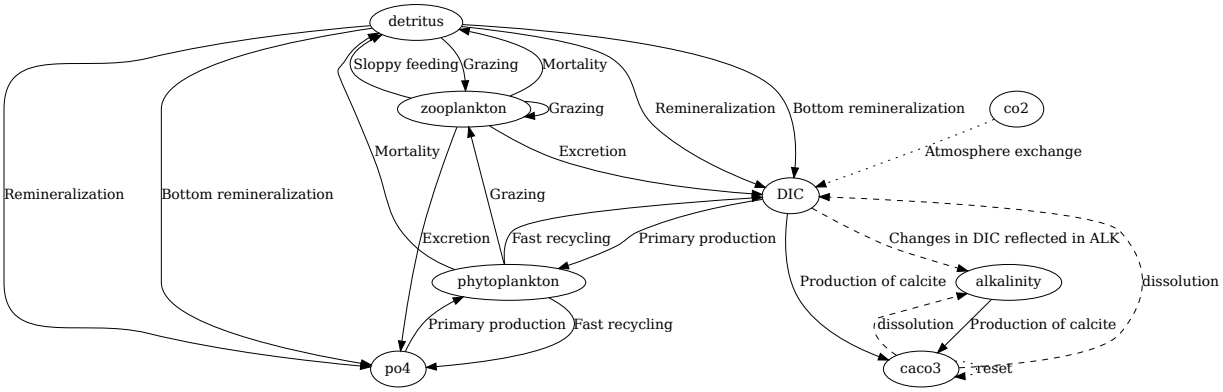


Figure 3: Auto-generated graph of the implemented dynamics of the carbon cycle

5 CO_2 exchange between ocean and atmosphere

In this section I will describe part of the global carbon cycle relating to CO_2 gas exchange between the ocean and atmosphere. The section includes estimates of equilibrium constants for multiple chemical reactions, which are calculated using recommendations based on experimental data. The sources do not provide accurate measures but rather approximations which assume a certain set of units. I will continue this assumption but make a note when appropriate. The units used are for temperature degrees Kelvin, for salinity PSU, for equilibrium constants $\mathrm{mol\,kg^{-1}}$.

One part of the global carbon cycle is the exchange of carbon dioxide between the ocean surface layer and the atmosphere. The transfer rate is dependent on the difference of partial pressure of CO_2 in the ocean and atmosphere. When solluted in water the carbon dioxide takes part in the ocean carbon chemistry resulting in a equilibrium problem, which can be solved numerically (Dickson and Goyet, 1994).

5.1 Gas exchange

The flux of CO_2 over the ocean-atmosphere boundary may be represented by the gas transfer coefficient relating the partial pressure difference of CO_2 in the ocean and atmosphere (Wanninkhof, 1992).

$$F = k\left(p\mathrm{CO}_{2\mathrm{water}} - p\mathrm{CO}_{2\mathrm{air}}\right) \tag{26}$$

The gas transfer coefficient, k, is a measure of transfer speed over the ocean-air interface. It is often estimated from the square of the mean wind speed, because it fits experimental results reasonably well, although there is not an intimate link between gas transfer rate and wind speeds (Wanninkhof, 1992).

$$k = 0.337 \cdot 0.75u^2 \left(Sc/660\right)^{-1/2} \tag{27}$$

The first factor in equation (27) is a scaling factor, u is the average wind speed and Sc is the Schmidt number. 660 is the Schmidt number for CO_2 in seawater at 20 °C.

The Schmidt number is defined as the kinematic viscosity of water divided by the diffusion coefficient of the gas. The value is estimated from a polynomial in sea surface temperature, T, in degrees Celsius with coefficients as presented by (Wanninkhof, 1992).

$$\mathrm{Sc}(\mathrm{CO_2}) \approx 2073.1 - 125.62T + 3.6276T^2 - 0.043219T^3 \tag{28}$$

With a description of how to calculate the flux of CO_2 gas over the ocean-atmosphere interface, the partial pressure $p\mathrm{CO_2}$ in the atmosphere and the ocean should be calculated. In the ocean, it can often be difficult to directly measure the concentration of aqueous CO_2 as it quickly establishes an equilibrium with H_2CO_3 displaying similar properties making it hard to distinguish them(Dickson and Goyet, 1994). They are therefore grouped together as $[\mathrm{CO_2}^\star]$. CO_2 concentration in the atmosphere is commonly referenced in units of ppmv and is the unit used by Veros, which isn't directly comparable to $[\mathrm{CO_2}^\star]$ and must therefore be converted to a number comparable to $[\mathrm{CO_2}^\star]$. This may be done like equation (29) as described in (Andreas Schmittner, 2018; Weiss, 1974).

$$[\mathrm{CO_2}^\star]_{(\mathrm{g})} = [\mathrm{CO_2}]_{\mathrm{ppmv}} \cdot f \cdot \mathrm{P_{atm}} \tag{29}$$

With P_{atm} the atmospheric pressure and f estimated by

$$\begin{aligned} \ln f = &-162.8301 + \frac{218.2968}{T/100} + 90.9241 \ln\left(T/100\right) - 1.47696 \left(T/100\right)^2 \\ &+ S\left(0.025695 - 0.025225\left(T/100\right)\right) + 0.0049867 \left(T/100\right)^2 \end{aligned} \tag{30}$$

It is not feasible to track the concentrations of each individual species of the carbon dioxide system for performance reasons, and because they can be difficult to measure directly. It is however possible to obtain a complete description of the carbon dioxide system knowing just 2 of 4 parameters: Total dissolved inorganic carbon, alkalinity, fugacity of CO_2 in equilibrium with sea water and total hydrogen ion concentration (Dickson and Goyet, 1994)

5.2 Ocean carbon chemistry

The oceanic carbon chemistry consists for the purpose of this work of a series of equilibria, which must be reached simultaneously in order to have a proper description of the CO_2 dissolved in the ocean.

When gaseous CO_2 is dissolved in seawater it reacts with water to form carbonic acid, which then dissociates to form bicarbonate and carbonate in the reactions described by equations (32) and (34)

$$CO_{2(g)} \rightleftharpoons CO_{2(aq)} \tag{31}$$
$$CO_{2(aq)} + H_2O_{(l)} \rightleftharpoons H_2CO_{3(l)} \tag{32}$$
$$H_2CO_{3(aq)} \rightleftharpoons H_{(aq)}{}^{+} + HCO_3{}^{-}{}_{(aq)} \tag{33}$$
$$HCO_3{}^{-}{}_{(aq)} \rightleftharpoons H_{(aq)}{}^{+} + CO_3{}^{2-}{}_{(aq)} \tag{34}$$

As it is difficult to analytically distinguish between $CO_{2(aq)}$ and $H_2CO_{3(aq)}$ it is convenient to group them together as $CO_2{}^{\star}$. Redefining equations equations (31) to (33) in terms of $CO_2{}^{\star}$. Which have the following equilibrium relationships:

$$k_0 = \frac{[CO_2{}^{\star}]}{f(CO_2)} \tag{35}$$
$$k_1 = \frac{[H^+][HCO_3{}^-]}{[CO_2{}^{\star}]} \tag{36}$$
$$k_2 = \frac{[H^+][CO_3{}^{2-}]}{[HCO_3{}^-]} \tag{37}$$

With $f(CO_2)$ the fugacity of atmospheric CO_2. Fugacity is related to partial pressure, but takes the non-ideality of the gas into account.

Neither of the unknowns in the equilibrium equations mentioned are being tracked in the model because they are non-conservative (Dickson and Goyet, 1994). However the model does track dissolved inorganic carbon as defined by equation (38) and total alkalinity, which is defined as the number of moles of hydrogen ion equivalent to the excess of proton acceptors over proton donors in one kilogram sample (Dickson and Goyet, 1994)

$$DIC = [CO_2{}^{\star}] + [HCO_3{}^-] + [CO_3{}^{2-}] \tag{38}$$

With $[OH^-]$ the concentration of the hydroxide ion and $[B(OH)^{4-}]$ is the concentration of the borate ion.

$$\begin{aligned} Alk = {} & [HCO_3{}^-] + 2\,[CO_3{}^{2-}] + [B(OH)_4{}^-] + [OH^-] + [HPO_4{}^{2-}] + 2\,[PO_4{}^{3-}] \\ & + [SiO(OH)_3{}^-] + [NH_3] + [HS^-] - [H^+] - [HSO_4{}^-] \\ & - [HF] - [H_3PO_4] + \text{minor acid or base species} \end{aligned} \tag{39}$$

The concentrations of borate, $[B(OH)_4{}^-]$, sulfate, $[HSO_4{}^-]$, flouride, $[HF]$ may all be estimated by the salinity (Dickson and Goyet, 1994)

Silicate, $[SiO(OH)^{3-}]$, and phosphorous ions $[HPO_4{}^{2-}]$, $[PO_4{}^{3-}]$, $[H_3PO_4]$ are assumed to be constant for the simulations described here (Andreas Schmittner, 2018).

The equilibrium balances which must be reached for the balance between DIC and alkalinity to be achieved are listed below. Above the balance arrows, I have listed the corresponding equilibrium constants.

$$[CO_2] \overset{k_1}{\rightleftharpoons} [H^+] + [HCO_3^-]$$
$$[HCO_3^-] \overset{k_2}{\rightleftharpoons} [H^+] + [CO_3^{2-}]$$
$$[H_3PO_4] \overset{k_1p}{\rightleftharpoons} [H^+] + [H_2PO_4^-]$$
$$[H_2PO_4^-] \overset{k_2p}{\rightleftharpoons} [H^+] + [HPO_3]$$
$$[HPO_4^{2-}] \overset{k_3p}{\rightleftharpoons} [H^+] + [PO_4]$$
$$[Si(OH)_4^-] \overset{ksi}{\rightleftharpoons} [H^+] + [SiO(OH)_3]$$
$$[H_2O] \overset{kw}{\rightleftharpoons} [H^+] + [OH^-]$$
$$[HSO_4] \overset{ks}{\rightleftharpoons} [H^+] + [SO_4]$$
$$[HF] \overset{kf}{\rightleftharpoons} [H^+] + [F]$$
$$[HBO_2] \overset{kb}{\rightleftharpoons} [H^+] + [BO_2]$$

If the equilibrium constants for the equilibria above are known, it is possible to express the equilibrium concentrations of the terms in equation (39) as functions of the $[H^+]$ concentration, which I will later present how to determine numerically. The concentrations in the alkalinity equation may then be determined as below (Dickson and Goyet, 1994).

$$[\mathrm{HCO_3{-}}] = \frac{\mathrm{DIC} \cdot \mathrm{k_1}\,[\mathrm{H^+}]}{[\mathrm{H^+}]^2 + \mathrm{k_1}[\mathrm{H^+}] + \mathrm{k_1}\,\mathrm{k_2}} \tag{40}$$

$$[{\mathrm{CO_3}}^{2-}] = \frac{\mathrm{DIC}\ \cdot \mathrm{k_1}\,\mathrm{k_2}}{[\mathrm{H^+}]^2 + \mathrm{k_1}\,[\mathrm{H^+}]\,\mathrm{k_1}\,\mathrm{k_2}} \tag{41}$$

$$[\mathrm{B(OH)_4{}^-}] = \frac{\mathrm{B_{total}}}{1 + [\mathrm{H^+}]\,/\,\mathrm{k_B}} \tag{42}$$

$$[\mathrm{OH^-}] = \frac{\mathrm{k}_w}{[\mathrm{H^+}]} \tag{43}$$

$$[\mathrm{H_3PO_4{}^-}] = \frac{\mathrm{P_{total}}[\mathrm{H^+}]^3}{[\mathrm{H^+}]^3 + \mathrm{k_{1}}_p\,[\mathrm{H^+}]^2 + \mathrm{k_{1}}_p\,\mathrm{k_{2}}_p\,[\mathrm{H^+}] + \mathrm{k_{1}}_p\,\mathrm{k_{2}}_p\,\mathrm{k_{3}}_p} \tag{44}$$

$$[\mathrm{H_2PO_4{}^-}] = \frac{\mathrm{P_{total}}\mathrm{k_{1}}_p\,[\mathrm{H^+}]^2}{[\mathrm{H^+}]^3 + \mathrm{k_{1}}_p[\mathrm{H^+}]^2 + \mathrm{k_{1}}_p\,\mathrm{k_{2}}_p[\mathrm{H^+}] + \mathrm{k_{1}}_p\,\mathrm{k_{2}}_p\,\mathrm{k_{3}}_p} \tag{45}$$

$$[{\mathrm{HPO_4}}^{2-}] = \frac{\mathrm{P_{total}}\mathrm{k_{1}}_p\,\mathrm{k_{2}}_p[\mathrm{H^+}]}{[\mathrm{H^+}]^3 + \mathrm{k_{1}}_p[\mathrm{H^+}]^2 + \mathrm{k_{1}}_p\mathrm{k_{2}}_p[\mathrm{H^+}] + \mathrm{k_{1}}_p\mathrm{k_{2}}_p\mathrm{k_{3}}_p} \tag{46}$$

$$[{\mathrm{PO_4}}^{3-}] = \frac{\mathrm{P_{total}}\mathrm{k_{1}}_p\mathrm{k_{2}}_p\mathrm{k_{3}}_p}{[\mathrm{H^+}]^3 + \mathrm{k_{1}}_p\,[\mathrm{H^+}]^2 + \mathrm{k_{1}}_p\mathrm{k_{2}}_p[\mathrm{H^+}] + \mathrm{k_{1}}_p\mathrm{k_{2}}_p\mathrm{k_{3}}_p} \tag{47}$$

$$[\mathrm{SiO(OH)_3{}^-}] = \frac{\mathrm{Si_{total}}}{1 + [\mathrm{H^+}]/\mathrm{k_{Si}}} \tag{48}$$

$$[\mathrm{HSO_4{}^-}] = \frac{\mathrm{S_{total}}}{1 + \mathrm{k_S}\,/\,[\mathrm{H^+}]_{\mathrm{free}}} \tag{49}$$

$$[\mathrm{HF}] = \frac{\mathrm{F_{total}}}{1 + \mathrm{k_F}\,/\,[\mathrm{H^+}]} \tag{50}$$

As mentioned B_{total}, S_{total} and F_{total} are assumed to be scaling with salinity(Morris and Riley, 1966; Uppström, 1974) and Si_{total}, P_{total} are assumed constant

$$B_{total} \approx 0.0023\frac{S}{10.811 \cdot 1.80655} \tag{51}$$

$$S_{total} \approx 0.14\frac{S}{96.062 \cdot 1.80655} \tag{52}$$

$$F_{total} \approx 0.000067\frac{S}{18.9984 \cdot 1.80655} \tag{53}$$

$$Si_{total} \approx 7.6875 \cdot 10^{-3} mol/m^3 \tag{54}$$

$$P_{total} \approx 0.5125 \cdot 10^{-3} mol/m^3 \tag{55}$$

It is possible to obtain a complete description of the carbon dioxide system given temperature and pressure, provided that the following is known:

- The solubility constant for CO_2 in sea water, $\mathrm{k_0}$

- The equilibrium constants for each of the acid/base pairs assumed to be present

- The total concentration of all non-CO_2 acid/base pairs

- The values for two of the CO_2 related parameters: total dissolved inorganic carbon, DIC, total alkalinity, Alk, fugcaity of CO_2, concentration of hydrogen ions, H^+ often given by $pH = -\log_{10}([H^+])$

The solubility constant for CO_2 in sea water, k_0, is related to the fugacity by

$$k_0 = \frac{[CO_2^\star]}{f(CO_2)} \tag{56}$$

Which may be estimated from temperature and salinity(Weiss, 1974)

$$\ln k_0 \approx \frac{93.4517}{T/100} - 60.2409 + 23.3585 + \ln(T/100) + S + (0.023517 - 0.023656 T/100 + 0.0047036 (T/100)^2) \tag{57}$$

Note that as mentioned in the beginning of this section, concentration is given in mol/kg, temperature is in kelvin, salinity in PSU.

The equilibrium constants for each of the acid/base pairs assumed present may be estimated from other tracked quantities. I have used the implementation from (Andreas Schmittner, 2018), which cites different sources for each estimated equilibrium constant. Some of the approximations rely on ionic strength in salt, IS, calculated by

$$IS = 19.924 S \frac{1}{1000 - 1.005 S} \tag{58}$$

$$\log_{10} \mathrm{k}_1 = \log_{10} \frac{[\mathrm{H}][\mathrm{HCO_3}]}{[\mathrm{H_2CO_3}]} \tag{59}$$

$$\approx -\left(3670.7/T - 62.088 + 9.7944 \ln T - 0.0118S + 0.000116S^2\right) \tag{60}$$

$$\log_{10} \mathrm{k}_2 = \log_{10} \frac{[\mathrm{H}][\mathrm{CO_3}]}{[\mathrm{HCO_3}]} \tag{61}$$

$$\approx -\left(1394.7/T + 4.777 - 0.0184S + 0.000118S^2\right) \tag{62}$$

$$\ln \mathrm{k}_{1p} = \ln \frac{[\mathrm{H}][\mathrm{H_2PO_4}]}{[\mathrm{H_3PO_4}]} \tag{63}$$

$$\approx -4576.752/T + 115.540 - 18.453 \ln T + (-106.736/T + 0.69171)\sqrt{S}$$
$$+ (-0.65643/T - 0.01844)S \tag{64}$$

$$\ln \mathrm{k}_{2p} = \ln \frac{[\mathrm{H}]\,[\mathrm{HPO_3}]}{[\mathrm{H_2PO_4}]} \tag{65}$$

$$\approx -8814.715/T + 172.1033 - 27.927 \ln T$$
$$+ (-160.340/T + 1.3566)\sqrt{S} + (0.37335/T - 0.05778)S \tag{66}$$

$$\ln \mathrm{k}_{3p} = \ln \frac{[\mathrm{H}]\,[\mathrm{PO_4}]}{[\mathrm{HPO_4}]} \tag{67}$$

$$\approx -3070.75/T - 18.126 + (17.27039/T + 2.81197)\sqrt{S}$$
$$+ (-44.99486/T - 0.09984)S \tag{68}$$

$$\ln \mathrm{k}_{\mathrm{Si}} = \ln \frac{[\mathrm{H}][\mathrm{SiO(OH)_3}]}{[\mathrm{SI(OH)_4}]} \tag{69}$$

$$\approx -8904.2/T + 117.400 - 19.334 \ln T + (-458.79/T + 3.5913)\sqrt{IS}$$
$$+ (188.74/T - 1.5998)IS + (-12.1652/T + 0.07871)IS^2 + \ln(1 - 0.001005S) \tag{70}$$

$$\ln \mathrm{k}_w = \ln [\mathrm{H}][\mathrm{OH}] \tag{71}$$

$$\approx -13847.26/T + 148.9802 - 23.6521 \ln T + 118.67/T - 5.977$$
$$+ 1.0495 \ln T\sqrt{S} - 0.01615S \tag{72}$$

$$\ln \mathrm{k}_{\mathrm{S}} = \ln \frac{[\mathrm{H}]\,[\mathrm{SO_4}]}{[\mathrm{HSO_4}]} \tag{73}$$

$$\approx -4276.1/T + 141.328 - 23.093 \ln T + (-13856/T + 324.57 - 47.986 \ln T)\sqrt{IS}$$
$$- 2698/TIS^{1.5} + 1776/TIS^2 + \ln(1 - 0.001005S) \tag{74}$$

$$\ln \mathrm{k}_{\mathrm{F}} = \ln \frac{[\mathrm{H}][\mathrm{F}]}{[\mathrm{HF}]} \tag{75}$$

$$\approx 1590.2/T - 12.641 + 1.525\sqrt{IS} + \ln(1 - 0.001005S) \tag{76}$$

$$\ln \mathrm{k}_{\mathrm{B}} = \ln \frac{[\mathrm{H}][\mathrm{BO_2}]}{[\mathrm{HBO_2}} \tag{77}$$

$$\approx (-8966.90 - 2890.53\sqrt{S} - 77.942S + 1.728S^{1.5} - 0.0996S^2)/T$$
$$+ (148.0248 + 137.1942\sqrt{S} + 1.62142S) + (-24.4344 - 25.085\sqrt{S} \tag{78}$$

Knowing total alkalinity and DIC, it is possible to calculate $[\mathrm{H}^+]$ by rearranging equation (39)

and solving the resulting equation in $[H^+]$ using a Newton-Raphson numerical solver. The average pH at sea surface is approximately 8. Hence a good first estimate for the solution is $[H^+] = 10^{-8}$.

When $[H^+]$ is known $[CO_2{}^\star]$ is calculated by

$$[CO_2{}^\star] = \frac{\mathrm{DIC} \cdot [H^+]^2}{[H^+]^2 + k_1[H^+] + k_1 k_2} \tag{79}$$

5.3 Numerical implementations

Having calculated the concentrations of relevant acid and base pairs, it is then left to solve the function in $[H^+]$ for which it is possible to analytically find a first derivate. I will first define the function to be optimized, f, and then its derivative in $[H^+]$ along with a few numbers to help simplifying the equations.

$$f = [HCO_3]+[CO_3]+[BH]+[OH]+[HPO_4]+2\,[PO_4]+[Si] - [H^+]_{\mathrm{free}} - [HSO_4] - [HF] - [H_3PO_4] - \mathrm{Alk} \tag{80}$$

$$\begin{aligned} a &= [H^+]^3\, k_{1p} + [H^+]^2\, k_{1p}k_{2p} + [H^+]\, k_{1p}k_{2p}k_{3p} \\ b &= [H^+]^2 + k_1\,[H^+] + k_{12}\,[H^+] \\ c &= 1 + \mathrm{Si}_{\mathrm{total}} \,/\, k_S + [HF] \,/\, k_F \end{aligned}$$

$$\begin{aligned} da &= 3\,[H^+]^3 + 2\,k_1\,[H^+] + k_1\, k\, 2 \\ db &= 2\,[H^+] + k_1 \end{aligned}$$

$$\begin{aligned} f =& k_1\,[H^+]\,\mathrm{DIC} \,/\, b + 2\,\mathrm{DIC}\,k_1\,k_2 \,/\, b + [B]_{\mathrm{total}} \,/\, (1 + [H^+] \,/\, k_B) + \\ & k_2 \,/\, [H^+] + P_{\mathrm{total}}\,k_{1p}k_{2p}\,[H^+] \,/\, a + 2\,P_{\mathrm{total}} \\ & k_{1p}k_{2p}k_{3p} \,/\, a + \mathrm{Si}_{\mathrm{total}} \,/\, (1 + [H^+] \,/\, k_{\mathrm{Si}}) - [H^+] \,/\, c - \\ & S_{\mathrm{total}} \,/\, (1 + k_S \,/\, ([H^+] \,/\, c)) - F_{\mathrm{total}} \,/\, (1 + k_F \,/\, ([H^+] \,/ \\ & c)) - P_{\mathrm{total}}\,[H^+]^3 \,/\, a - \mathrm{Alk} \end{aligned}$$

With the derivative, df, in $[H^+]$

$$\begin{aligned} df =& ((k_1\,\mathrm{DIC} \cdot b) - k_1\,[H^+]\,\mathrm{DIC}\,db) \,/\, b^2 - 2\,\mathrm{DIC}\,k_1\,k_2\,db \,/\, b^2 - \\ & B_{\mathrm{total}} \,/\, k_B \,/\, (1 + [H^+] \,/\, k_b)^2 - k_w \,/\, [H^+]^2 + (P_{\mathrm{total}} \\ & k_{1p}k_{2p}\,(a - [H^+]\,da)) \,/\, a^2 - 2\,P_{\mathrm{total}}\,k_{1p}k_{2p}k_{3p}\,da \,/\, a^2 \\ & - \mathrm{Si}_{\mathrm{total}} \,/\, k_{\mathrm{Si}} \,/\, (1 + [H^+] \,/\, k_{\mathrm{Si}})^2 - 1 \,/\, c - S_{\mathrm{total}} \,/\, (1 + k_S \,/ \\ & ([H^+] \,/\, c))^2\,(k_S\, c \,/\, [H^+]^2) - F_{\mathrm{total}} \,/\, (1 + k_F \,/\, ([H^+] \,/ c))^2\,(k_F\, c \\ & / \,[H^+]^2) - P_{\mathrm{total}}\,[H^+]^2\,(3\,a - [H^+]\,da) \,/\, a^2 \end{aligned}$$

The roots of f, are where a chemical balance will be found which allows determining $[CO_2^\star]$. To find the roots of f I employ a Newton-Raphson solver with bisection within a bounded interval following the reference of (Andreas Schmittner, 2018). Whenever the Newton-Raphson step would take the solution out of bounds or if the Newton-Raphson is not reducing the size of the solution bound sufficiently fast the solver takes a bisection step. The implementation is based on the rtsafe function from Numerical Recipes (Press, 2002). Equations (81) and (82) are used to determine if a potential solution is out of bounds or not decreasing fast enough respectively.

The solver first calculates the function values at the endpoints and determines which is the lower bound, and which is the higher by orienting the search such that the function value at the lower bound is negative. There is no error handling for the case where both endpoints have positive or both negative values as the input should be provided by the model with safe values. In the rtsafe implementation from Numerical Recipes, the first guess for a solution, is halfway between the endpoints. I use the same guess for the first iteration during the first time step, but any subsequent guesses will be the solution from the previous calculation. By using this technique I have reduced the number of iterations required to find a solution with the desired accuracy. An iteration consists of the following steps: The function value is determined. If the resulting solution would be out of range, or the absolute function value isn't decreasing fast enough, the corresponding step would be a bisection with a step size half the distance between the endpoints. If the Newton-Raphson step is within range, it is performed. Then the step size is compared to a predetermined accuracy. If the step size is smaller, it found an acceptable solution. If not, adjust the endpoints such that if the function value of the current attempted solution is above zero, set the function value as the upper limit. Otherwise set the function value as the lower limit. Then start a new iteration.

When a solution has been found, it is a reasonable estimate of the solution at the next time step. Therefore I reuse the solution as the initial guess and set the endpoints corresponding to the shortest distance to either of the two initial assumed safe boundaries but keeping a minimum boundary width. The implementation in UVic ESCM keeps the boundaries at $\pm$05pH of the initial guess. In section 12 I explain the reason for implementing a different solution.

Some of the cells in the data grid are land. Therefore calculating the flux of CO_2 between the atmosphere and ocean surface could be pointless and may even be expensive computationwise. Therefore a mask is applied to the array to ensure, the solver only searches for solutions where it matters. Furthermore, the mask is updated dynamically, such that the solver stops calculating in cells, where a solution has already been found.

To determine whether the solution at x is out of bounds set by x_{low} to x_{high} the following is tested

$$((x - x_{high})df(x) - f(x)) \cdot ((x - x_{low}) \cdot df(x) - f(x)) > 0 \tag{81}$$

Another reason to bisect may be, if the absolute function value isn't decreasing fast enough determined by the condition

$$|2 \cdot f(x)| > |dx_{old} \cdot df(x)| \tag{82}$$

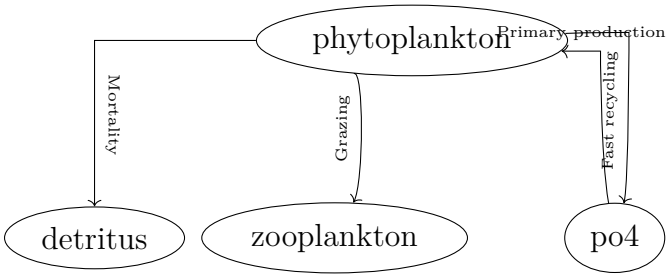
6 Rule based model

In sections 2 and 3 I covered the dynamics of the ocean biogeochemical system. I argued that an increase in the concentration of one tracer can be combined with a decrease in another. In some cases the transaction is split up, such as when zooplankton consumes phytoplankton, the zooplankton grows but also excretes nutrients and some of the consumed phytoplankton is lost to sloppy feeding as detritus, but there is a flow from the phytoplankton tracer to the zooplankton tracer and other flows from the zooplankton to nutrients and detritus. For every interaction between tracers it is possible to define a flow of concentration from one tracer to another. Such a flow corresponds to each arrow in figure 1.

I showed, that it is possible to define a function between any two interacting tracers, which returns the change in each tracer due to the dynamic described by the function. That means, that for each term in the source-minus-sink equation for any of the tracers in the model, it is possible to define a function returning the decrease in the source and increase in the sink. I define the tuple consisting of the function, name of the source, and name of the sink as a rule. By defining rules for every node in the interaction graph in figure 1, I have simultaneously created terms in the source-minus-sink equations for the nodes connected by the edge. In the example below, I am demonstrating, how to construct rules for the interaction set relating to phytoplankton in the basic NPZD model.

Example 1: Using rules to build dynamics

The graph below shows the interaction set for phytoplankton - a subset of the dynamics in a complete model.



The same dynamics may be expressed with the source-minus-sink equations. The labels above each term correspond to a label on the graph above.

$$
\begin{aligned}
S(P) &= -\overset{Grazing}{G(P)} - \overset{Mortality}{\mu_P P} - \overset{Fast\_recycling}{\mu_{Pt} P} + \overset{Primary\_production}{JP} \\
S(\mathrm{PO_4}) &= \overset{Primary\_production}{-R_{P:N} JP} + \overset{Fast\_recycling}{R_{P:N}\mu_{Pt} P} \\
S(Z) &= \overset{Grazing}{\gamma_1 G(P)} \\
S(D) &= \overset{Mortality}{\mu_P P}
\end{aligned}
$$

This may be built in the model by creating functions representing the edges of the graph.

```
@veros_method(inline=True)
def grazing(vs, eaten, zooplankton):
    """ Zooplankton grows by amount digested, eaten decreases by amount grazed """
    return {eaten: - vs.grazing[eaten], zooplankton: vs.grazing[eaten]}

@veros_method(inline=True)
def mortality(vs, plankton, detritus):
    """ All dead matter from plankton is converted to detritus """
    return {plankton: - vs.mortality[plankton], detritus: vs.mortality[plankton]}

@veros_method(inline=True)
def recycling(vs, plankton, nutrient):
    """ plankton or detritus is recycled into nutrients """
    return {nutrient: vs.redfield_ratio_PN * vs.recycled[plankton], plankton: - vs.
        recycled[plankton]}

@veros_method(inline=True)
def primary_production(vs, nutrient, plankton):
    """ Primary production: Growth by consumption of light and nutrients """
    return {nutrient: - vs.redfield_ratio_PN * vs.net_primary_production[plankton],
        plankton: vs.net_primary_production[plankton]}

register_npzd_rule(vs, 'example_grazing', (grazing, 'phytoplankton', 'zooplankton'),
    label='Grazing')
register_npzd_rule(vs, 'example_mortality', (mortality, 'phytoplankton', 'detritus'),
     label='Mortality')
register_npzd_rule(vs, 'example_fastrecycling', (recycling, 'phytoplankton', 'po4'),
    label='Fast recycling')
register_npzd_rule(vs, 'example_primary_production', (primary_production, 'po4', '
    phytoplankton'), label='Primary production')
```

Every rule corresponds to an edge in the graph. It consist of a source and a sink and a function relating growth in the sink to the decrease in the source. As rules are selected in Veros, the graph and equation set is built by the contents of the rule.

Example 1 introduces the concept of registering rules. Separating the registration of rules from activation of rules forms the basis of the flexibility and ease of use of the model. The process of adding rules, thereby changing the behaviour of the model should be doable without knowledge of how other users are configuring their model setups. Furthermore any rule could be used in several configurations. Therefore when defining a new rule, it is made available in a list containing all rules. When a user wants to include a rule, it must be explicitly selected in the setup configuration file. An extension modifying the behaviour of an existing rule would therefore not edit the existing rule directly but rather write a new rule specifying the new behaviour. Since the results of a rule are calculated in a single function, simple extensions of a rule may choose to call the original function and provide its own calculations on top. A more complex edit for defining significantly modified behaviour should define a new function for specifying its behaviour. By collecting all defined rules in a single collection and separately specifying which rules to activate, I have facilitated constructing highly specialized configurations and general behaviour in the same code base without reducing readability of the control structure or any single component of the dynamics. This is because the execution of rules is done independently of every other rule at fixed locations, which contributers writing rules will not be required to modify. In order for rules to provide adequate flexibility, I have added the option to specify

during the registration of rules, where in the grid the rule should apply to as well as when in the execution order to apply the rule. Not in terms of ordering execution of rules in time, but to specify if the rule should be applied before the main loop, during or after. Figure 4 shows the difference between registered rules and selected rules.

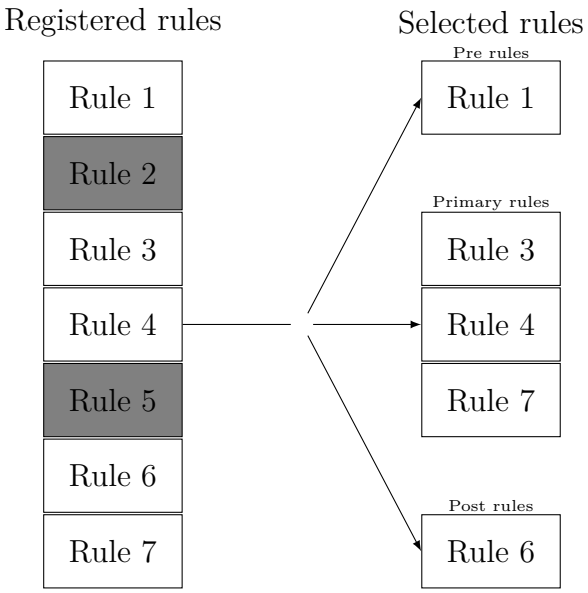


Figure 4: On the left are all rules, which are registered in Veros. Rules with gray background were not selected for activation. Upon starting the biogeochemistry module, selected rules are separated into execution groups. Each group is applied at different points during model evaluation.

When a rule is applied, it is done so at the point specified during registration. The result of every rule in the execution group is evaluated before adding the result to tracer values as shown in listing 9. I will expand upon the use of execution groups in section 7.

Listing 9: Applying rules

```
# Each rule group is evaluated like
rule_results = [(rule.function(vs, rule.source, rule.sink), rule.boundary) for rule in vs.npzd_rules]

# Then the tracers are updated
for rule, boundary in rule_results:
    for tracer, update in rule.items():
        vs.npzd_tracers[tracer][boundary] += update
```

A rule as described contains four entities: A name, a function, the name of the source, and the name of the sink. But as shown in listing 10, it is possible to add additional information. A rule can specify an execution group. The execution group is identifying, when in the execution order, the rule is evaluated. A rule can belong to the primary execution group. This is the default, which is evaluated at every time step for biogeochemistry. This group has access to the most recently updated values of diagnostic parameters, and determines the primary interaction of the system. Rules may also specify membership of the PRE or POST groups. PRE rules are evaluated before the evaluation of time steps, and the POST rules are evaluated afterwards.

The PRIMARY rules should return time derivative updates and follow the source-minus-sink descriptions of previous sections, whereas the PRE and POST rules are intended for smoothing or clean-up and are therefore better suited as absolute additions.

The boundary parameter specifies the cells the rule applies to. For example the SURFACE boundary is useful in implementing atmosphere exchanges, since the results of such a rule is added to the surface layer. The BOTTOM boundary applies the results to the cells in the ocean bottom layer, not the bottom grid layer. If no boundary is specified, the rule applies to every cell. Internally the boundaries are implemented as python slices. If an extension requires working on a different boundary, a new slice or mask can be created and added to the python dictionary storing boundaries. I decided against allowing setting the boundary slice directly in the rule creation to keep the rule creation format consistent and to have a single location to look up available boundaries. Finally, rules can set a label. This label is not an identifier, but is shown in the auto generated graphs. I chose to construct the rules as named tuples for two reasons. One is it makes the code more readable to read rule.function(rule.source, rule.sink) compared to rule[0](rule[1], rule[2]). The other is, it allows for extending rule tuples to contain more fields without worrying about order.

In listing 10 I provide a full example of registering a rule in Veros' biogeochemistry module. The example uses a template to represent common functionality for calcite production by different species, which is used in the actual rule function. The rule is then registered with a name, function, source, sink, label, execution group and boundary to explicitly state the intended behaviour. Finally the rule is selected in the npzd.yaml configuration file.

Listing 10: Defining and selecting an extensible rule

```
# npzd_rules.py
# This rule is a template for calcite production with different plankton types
# or detritus
@veros_method(inline=True)
def calcite_production(vs, plankton, DIC, calcite):
    """ Calcite is produced at a rate similar to detritus
        Intended for use with a smoothing rule
        If explicit tracking of calcite is desired use
        rules for the explicit relationship
    """

    # changes to production of calcite
    dprca = (vs.mortality[plankton] + vs.grazing[plankton] *
            (1 - vs.assimilation_efficiency)) * vs.capr * vs.redfield_ratio_CN

    return {DIC: -dprca, calcite: dprca}


# This rule uses the template to calculate the explicit calcite production by
# phytoplankton
@veros_method(inline=True)
def calcite_production_phyto(vs, DIC, calcite):
    """ DIC is consumed to produce calcite. How much depends on the mortality
        and sloppy feeding of and on phytoplankton and zooplankton
    """
    return calcite_production(vs, 'phytoplankton', DIC, calcite)

# npzd.py
# Register the rule along with when and where to execute it
register_npzd_rule(vs, 'npzd_carbon_calcite_production_dic', # selectable name
                (calcite_production_phyto, 'DIC', 'caco3'), # (function, source, sink)
                label='Production of calcite', # optional but recommended for graph
                boundary='NONE', # optional, SURFACE, BOTTOM or everything
                group='PRIMARY') # optional, default is 'PRIMARY'

# npzd.yaml
# Select the rule for use
npzd:
  selected_rules:
    - "npzd_carbon_calcite_production_dic"
    ...
  ...
```

For ease of use and in order to categorize related rules, Veros allows specifying rule collections as a single rule containing related rules, which allows the user to get a tested configuration without having to pick each individual rule. Once a set of rules has been tested and found to be working correctly, they should not change between model setups intending to use the same configuration. Therefore I recommend creating a configuration file with the selected configuration as in listings 11 and 12. This configuration is portable between model setups, such as developing locally on a low resolution model and running a higher resolution model with the same dynamics on a more powerful machine. If the user does not want to manage multiple files,

it is also possible to specify the selected rules directly in the launch file for the Veros simulation like any other model variable by specifying the contents of the list vs.selected_tracers. A configuration file, which selects rules for a basic NPZD simulation and additional rules for a carbon cycle looks like:

Listing 11: Rule setup for carbon cycle using rules

```
npzd:
  selected_rules:
    - 'group_npzd_basic'
    - 'group_npzd_carbon'
```

The same rules can be selected individually by selecting each by name. Selecting collections of rules allows for easy to manage configuration files. Selecting each rule individually allows for full, explicit control of which rules are activated. Which is especially useful during model development, as it allows for easy activation/deactivation of individual rules.

Listing 12: Rule setup with explicit rule selection

```
npzd:
  selected_rules:
    - 'npzd_basic_phytoplankton_grazing',
    - 'npzd_basic_phytoplankton_mortality',
    - 'npzd_basic_phytoplankton_fast_recycling',
    - 'npzd_basic_phytoplankton_primary_production',
    - 'npzd_basic_zooplankton_grazing',
    - 'npzd_basic_zooplankton_excretion',
    - 'npzd_basic_zooplankton_mortality',
    - 'npzd_basic_zooplankton_sloppy_feeding',
    - 'npzd_basic_detritus_remineralization',
    - 'npzd_basic_detritus_grazing',
    - 'npzd_basic_detritus_bottom_remineralization'
    - 'npzd_carbon_flux',
    - 'npzd_carbon_recycling_detritus_dic',
    - 'npzd_carbon_primary_production_dic',
    - 'npzd_carbon_recycling_phyto_dic',
    - 'npzd_carbon_excretion_dic',
    - 'npzd_carbon_dic_alk',
    - 'npzd_carbon_calcite_production_dic',
    - 'npzd_carbon_calcite_production_alk',
    - 'npzd_carbon_post_distribute_calcite_alk',
    - 'npzd_carbon_post_distribute_calcite_dic',
    - 'npzd_carbon_detritus_bottom_remineralization',
    - 'pre_reset_calcite',
```

A model design like UVic ESCM would calculate the terms of the equations of dynamics individually and then add them up in equations like equation (3). A rules based model defines a collection of rules or functions over which it iterates, evaluating the rules and adding the results to the tracers defined by the rule to produce the same result.

When multiple configuration options are available, it is necessary to check which combination of options have been selected in order to construct the correct source-minus-sink equations. As the number of configuration options grows, so does the amount of combinations of configurations,

which makes the code base difficult to maintain and reduces readability. By defining an iterable collection of rules, the control structure remains the same. The rules are defined once, separate from the flow of evaluation. Each rule is evaluated once and may be inspected separately from the selected configuration combination.

The tracers in the biogeochemistry module of Veros share some common behaviour. Every tracer must be set to have a minimum value at every time step in order to avoid negative concentrations and some tracers sink from one vertical level to the one below. To accommodate the similarity a reference to every biogeochemistry tracer is stored in a Python dictionary, which is iterable allowing for common procedures on the tracers. Before registering the tracers, they are defined as objects, which contain the concentration in every cell of the grid as well as additional information about the tracer. This is explained in further detail in section 8. When adding a new tracer to the model, it is then only required to register the tracer in the NPZD tracer collection. The registered tracer is then automatically transported by advection and diffusion, it is always ensured a minimum concentration, and the tracer is shown by name in an auto generated interaction graph like figure 3. Common operations applied to a subset of tracer collection like sinking or absorbing light may be performed by iteration over the tracers relating to that operation. This approach and its advantages and alternatives are described in further detail in section 8.

The list of all rules fully describes the dynamics of the biogeochemistry module with the exclusion of tracer transport. The list of rules for a basic NPZD model like the one described in section 2 may be extended by rules corresponding to the description in section 4 to allow for a carbon cycle or it could be extended with rules for a calcium cycle or both.

As more configurations are needed, the user is left with a tree of conditions for which rules should be active, leaving only a slightly improved situation from the model design I intended to improve upon. The chosen solution is to store all available rules in a single list and allow the user to provide a list with identifiers for the rules to activate. A complex configuration would contain a large number of active rules. Therefore I also allow collections of rules to be activated. Once a collection has been tested and is expected to be reused, it may be stored as a list of the active rules. Users wanting to build models extending the saved configuration then select the rule group and any additional rules.

Veros provides configuration for common model setups like a basic NPZD model with phosphate as the only nutrient, and a carbon cycle with atmosphere-ocean gas exchange. Each can be enabled selecting the rule group and setting a flag to add the required tracers.

It is even possible to inject rules or tracers during execution by appending to the lists should it be required, although I discourage this practice, because it obscures the modelled behavior.

7 Explicit and smoothed representation

In section 6 I described how the rule structure of the biogeochemistry module in Veros works. The described structure demands that any change to a tracer should be fully describable by one rule with a distinct source and sink. It has also been assumed, that the process described by the rule is contained within the same grid cell. The biogeochemistry module has three places in which to add rules: Pre rules, inner rules and post rules. The pre rules are executed first. The pre rule set is intended for updates, that only need to occur once per tracer time step and has to happen before the other rules. An example of such a rule could be carbon exchange between the ocean surface layer and the atmosphere. If the model does not explicitly track the

concentration of calcium carbonate, then one evaluation of the exchange per tracer time step may prove sufficient.

The primary interaction rules should be placed in the inner rule set. The rules placed here will be performed a number of times, n_{bio}, based on the selection of time step length defined for the simulation. In a closed, explicitly described system, the rules in the inner rule set should give a complete description of the biogeochemical dynamics.

When the execution of the inner rules complete, the final rule set, the post rules, are activated once. These rules define actions, which should occur only once per time step and must happen after the dynamics of the overall tracer time step are completed. For example I described a rule in section 2 where detritus should be remineralized as it falls through the ocean bottom. This could be handled in the inner rules, but a configuration could require the detritus to stay at the bottom until completion of the time step at which point it is remineralized. By defining a post rule rather than an inner rule, it can collect the detritus at each step in the primary execution loop until the completion of the overall tracer time step and then remineralize the detritus to nutrients.

As noted in the beginning of this section any interaction between tracers has thus far required a distinct source and sink and a single function describing the flow between them. It is however possible to not define the flow from one tracer in one grid cell to another tracer in the same cell but to collect a contribution by multiple sources which is then redistributed by a predetermined statistical measure throughout the grid smoothing the gathered state update. This type of interaction representation will be called smoothing representation

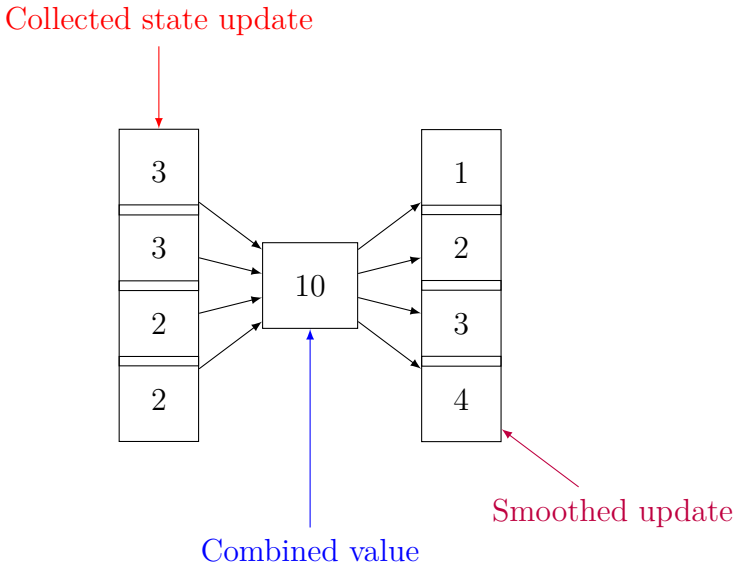


Figure 5: State updates in a vertical column may be smoothed over the entire column using a smoothing rule

The default rule configurations provided by Veros have been written to use smoothing rules in the carbon cycle. In the carbon cycle $CaCO_3$ contributes to the alkalinity of the ocean and dissolved inorganic carbon through production and dissolution as described in section 4. It is possible to explicitly represent $CaCO_3$ as a tracer and updates to it as rules, however if an experiment is only interested in the dissolution and production and the effect on existing tracers in the model, it may be convenient to leave out a full description of calcite and only store a variable to collect state updates by dissolution and production. Without explicitly tracking the

calcite throughout the cycle, the collected state update must be redistributed by a predefined measure. This is a smoothing representation.

For the actual representation in Veros, I defined a tracer to collect the state updates and registered it in the tracers collection, as I would any other tracer, but I set a flag, such that it is not included in the physical transport, saving some compute time. I then defined rules for the production of calcite consuming DIC and alkalinity. These rules were added to the inner rule set, such that calcite is produced at every bio time step. I then defined two additional rules describing the dissolution of $CaCO_3$ into DIC and alkalinity. These rules sum up the produced $CaCO_3$ in vertical columns and redistribute it vertically according to the description in equation (21). The rules were added to the post rule set. By this procedure $CaCO_3$ will build up during the internal dynamics and is then redistributed afterwards. Finally I added a rule to the pre rules set in order to reset the $CaCO_3$ state tracer. Although recent versions of Python guarantee dictionary order is the same as insertion order, it is better to explicitly reset the state tracer than to rely on a rule implicitly resetting it.

When creating a smoothing rule, which consists of several individual rules, it should be created as a collection of rules, because the defined functionality only works for the collective behaviour of all rules in the collection. If other users were to use the smoothing rule. They should only be aware of the overall purpose of the smoothing rule, not its components.

Listing 13: Register a smoothing rule as a rule group

```
register_npzd_rule(vs, 'smoothing_calcite_production',
                   ['pre_reset_calcite',
                   'npzd_carbon_calcite_production_dic',
                   'npzd_carbon_calcite_production_alk',
                   'npzd_carbon_post_distribute_calcite_dic',
                   'npzd_carbon_post_distribute_calcite_alk'])
# Registering rules groups does not accept labels, boundaries or groups
# That is left for the individual rules
```

By defining a smoothing rule, I have avoided requiring a complete description of $CaCO_3$ in order to use its effect on the carbon cycle.

8 Object Representation

In the previous sections, I mentioned that every tracer has a set of properties related to it such as mortality rate for plankton or whether or not it blocks light or it is necessary to transport the tracer i.e. whether it is a temporary storage tracer such as $CaCO_3$ in the configuration without explicit calcifiers I used in section 4.

As mentioned in section 6 every tracer is stored in a Python dictionary, which allows accessing it by name. This can be useful, when calculating variables, which may be used in multiple rules. For example to calculate the amount of recycled material of all tracers, it would be sufficient to iterate over a collection of recycling rates and multiply it onto the concentrations of the corresponding tracers. However, handling cases which should not follow the convention set by other tracers becomes increasingly difficult as the number of tracers increase and requirements diverge. An example of such an exception is zooplankton mortality. Other plankton types display mortality scaling linearly with their concentrations. Zooplankton on the other hand is modelled with quadratic increase in mortality which would require additional configuration in the logical structure of the model calculations. Creating classes for the tracers allows representing different behavioural patterns in mortality, primary production and recycled tracers by implementing methods differently while inheriting common behaviour. Tracers which behave with a common pattern but at different rates may simply be instantiated with different rate parameters. This is exemplified in listing 14.

Variables in Veros are stored in multidimensional Numpy arrays. The tracers in the biogeochemistry module are no exception to this. The base tracer class inherits from numpy.ndarray which makes it interoperable with any regular Numpy array for array operations and allows it to be used in accelerated calculations with the Bohrium backend without any additional modification. The new tracers are different in that they carry attributes and may implement methods for the pre-calculations of results for use in rules. In order to make use of the diagnostics provided by Veros for snapshotting and averaging, any biogeochemistry tracer should be instantiated with a Veros variable. The tracer object then acts as a view onto the variable allowing complete interoperability with other parts of Veros such as the diagnostics modules.

By creating classes I have effectively created a container for storing all relevant information about a tracer, which may easily be created, retrieved and extended. The example in listing 14 demonstrates extending an existing Plankton class by overriding the mortality method as well as instantiating similar objects with different parameters. The simplest class, NPZDObject, defines attributes for whether or not to transport the tracer which defaults to True, a light attenuation factor with default value 0, and an option to define a sinking speed. The Plankton class inherits from the NPZDObject class and defines methods for mortality and recycling. Inheritance allows for easily creating different model tracers, and focusing on the differences, which makes the tracer unique.

Listing 14: Creating and extending model tracers

```
class Zooplankton(Plankton):
  """ Zooplankton is like Plankton but with quadratic mortality rate and grazing"""

  # __init__ not shown

  @veros_method(inline=True)
  def mortality(self):
    return self.mortality_rate * self ** 2

  # Grazing method not shown

# Instatiate the zooplankton object from the created Zooplankton class
zooplankton = Zooplankton(vs.zooplankton, 'zooplankton',
                      max_grazing=vs.maximum_grazing_rate,
                      grazing_saturation_constant=vs.saturation_constant_Z_grazing,
                      assimilation_efficiency=vs.assimilation_efficiency,
                      growth_efficiency=vs.zooplankton_growth_efficiency,
                      grazing_preferences=vs.zprefs)

# Instatiate phytoplankton object
# Note, the Phytoplankton class is not shown. It defines a method for primary production
phytoplankton = Phytoplankton(vs.phytoplankton, 'phytoplankton',
                          light_attenuation=vs.light_attenuation_phytoplankton,
                          growth_parameter=vs.maximum_growth_rate_phyto,
                          recycling_rate=vs.fast_recycling_rate_phytoplankton,
                          mortality_rate=vs.specific_mortality_phytoplankton)

# Instatiate coccolithophore object which is similar to phytoplankton
# but with different recycling rate
cocco = Phytolankton(vs.coccolithophore, 'coccolithophore', # veros variable and name
                   light_attenuation=vs.light_attenuation_phytoplankton,
                   mortality_rate=vs.mortality_rate_phytoplankton,
                   recycling_rate=vs.recycling_rate_coccolithophore, # different rate
                   growth_parameter=vs.maximum_growth_rate_coccolithophore) # different
                       rate
```

The methods defined by the classes calculate numbers which are used in multiple rules. The methods follow a naming structure, which allows any object to define methods with one of these names. For example mortality for any tracer is calculated by checking if the tracer has a method named mortality, and if it does, it is called and the result is stored in a dictionary with key the name of the tracer. This approach has a slight overhead in checking tracers for attributes they may not have at every time step. The overhead is not considered large as the configurations described in this thesis do not manage a large amount of tracers, and the approach has been kept in order to preserve the readability of the code base. Should the overhead prove to be significant, it would be possible to check which tracers contain a certain method and store the names in a list during registration. When the model is running, it will then access the tracers by the names stored in the lists.

The biogeochemistry module features automatic generation of graphs displaying interactions between tracers by selected rules. Every rule defined may specify an optional description of itself, which is displayed on the graph. Similarly the tracer objects can store additional metadata

like a description of the object, a symbol or name to be used in the graphs or other auxiliary information.

8.1 Evaluation structure

With the concept of rules introduced in section 6 and the different points, they can be executed at, presented in section 7 along with the object methods from section 8 I have created building blocks for the biogeochemistry module. The biogeochemistry is split from the physical transport. They are calculated separately and the results are added to the tracer values for the next time step. The transport uses the general transport scheme in Veros, using advection and isopycnal diffusion. The biogeochemistry is calculated as follows:

1. Calculate biogeochemistry
    (a) Update variables which only depend on time dependent values from other Veros modules
    (b) Evaluate pre rules and update tracers
    (c) Ensure minimum concentrations of updated tracers and set flags
    (d) Enter biogeochemistry loop
        i. Evaluate object methods for primary production, mortality, recycling etc.
        ii. Reduce light availability by light attenuation factors and concentrations of tracers
        iii. Calculate import - export for sinking tracers - but don't update
        iv. Evaluate primary rules and update tracer values
        v. Update tracer values from import - export and remineralize tracers passing through the ocean bottom
        vi. Ensure minimum concentrations and set flags
    (e) Evaluate post rules and update tracers
    (f) Ensure minimum concentrations of updated tracers
2. Calculate physical transport which updates tracers in next time step
3. Add tracer updates from biogeochemistry calculations to next time step

In appendix B I provide a simplified code example showing the evaluation structure.

9 Design choices

The initial goal of this work was to reimplement the functionality of the npzd implementation of UVic ESCM written in Fortran (Andreas Schmittner, 2018) into Veros. UVic ESCM has been used in multiple publications with great model accuracy and remains a good option for simulating biogeochemistry while providing more functionality than Veros. This section describes why and how the biogeochemistry implementation in Veros has diverged from the UVic ESCM implementation. I will outline several previous iterations of the model design leading up to the final implementation described in previous sections. In describing each iteration, I will emphasize the need for an alternative design and my solution to the problem.

In the Fortran implementation in UVic ESCM, every tracer is defined as a single variable as well as every term contributing change to it. This means that the model contains several lines of code displaying similar functionality only with a parameter changed. The common approach to simplifying such a code base is to define functions, however as shown in section 6, the functions for common behaviour is exceedingly simple, once the common pre-calculated terms have been calculated. Defining functions for this, does not in itself reduce the complexity of the code base.

Another issue is that common for all tracers in the model is that they need to maintain a minimum concentration as well as a flag specifying whether the concentration has been reset in the current time step. This requires at least two lines of code per tracer at every location, the tracer needs to be reset. At the time of writing with the current functionality just ensuring minimum tracer concentration before evaluating the first time step in UVic ESCM as present at (Andreas Schmittner, 2018) requires 101 lines of code. The iterative approach in Veros, which I will describe below, requires 3 lines of code and remains at 3 lines even when adding additional tracers to the model. Similar arguments may be applied to calculations of grazing, primary production, mortality and recycling rates.

This lead to the first implementation of biogeochemistry in Veros. Every tracer is stored in a python dictionary. Therefore operations such as resetting tracer concentrations to a minimum value could be completed by iterating over the dictionary of tracers. The process of calculating grazing, mortality and recycling were similarly simplified by maintaining python dictionaries of the corresponding grazing preferences and rates for mortality and recycling as shown in listing 15. The dictionary approach has the advantage of making it easy to add new recyclable tracers or other functionality to the model without having to define the behaviour multiple times. This meant it was possible to separate which tracers were in the model from the evaluation structure. That is an advantage in terms of readability and maintainability, because the logical structure now only contains the description of an action once, and it does not need to check whether a certain configuration of tracers is active. Furthermore I gain the advantage of having each of the terms in the source-minus-sink equations stored in a common structure, which is made available to the diagnostics module and can be output to a file, again without having to check for active configurations.

Listing 15: Maintaining dictionaries for rates

```
vs.recycling_rates = {'detritus': vs.recycling_rate_detritus,
'phytoplankton': vs.recycling_rate_phytoplankton}

# The Dictionary can easily be extended if necessary
vs.recycling_rates['diazotroph'] = vs.recycling_rate_diazotroph

...

# Dictionaries are iterable. By storing rates we don't need to
# peform checks on the presence of a tracer at runtim
for tracer, rate_factor in vs.recycling_rates.items():
    vs.recycled[tracer] = rate_factor * vs.temporary_tracers[tracer]

...
```

However the described approach does not handle the possibility for tracers to display differing behaviour. For example as described in section 2, zooplankton is modelled with a quadratic mortality rate, whereas every other tracer is modelled with a linear mortality rate. The dictionary approach does not allow for such behaviour. With the configurations described in sections 2 and 4, the quadratic mortality rate for zooplankton could be handled by inserting a single line multiplying the recycling rate by the zooplankton concentration again.

Listing 16: Zooplankton mortality rate

```
vs.recycled['zooplankton'][...] *= vs.temporary_tracers['zooplankton']
```

Doing so does not slow down the model evaluation or make it harder to read, but it does show, that the described approach is not flexible enough to handle functionality, which would differ from the assumed general behaviour by more than just the value of factors. This became obvious when I extended the model by adding a nitrogen cycle and calcifiers. I have left out the description of those thus far, as they were not relevant to the experiment described in section 13, and the code has not been rewritten to support the rule based model described in section 6. In section 9.1 I will describe the possibility of adding such functionality back into Veros.

It should also be noted, that I in section 2 described the sinking of tracers and how they are remineralized at the bottom. That is the amount of the tracer, which would have sunk through the bottom of the cell at the ocean floor is remineralized into nutrients. There are two reasons why that does not fit well within the model setup which asserts common behaviour of tracers for all configurations. For one assuming the sinking amount relates to how much is remineralized at the bottom, is likely to change in more advanced setups. Secondly specifying which tracers the sinking material remineralizes to and in which amounts becomes hard to manage within the framework of iterating over dictionaries. It becomes necessary to consider the active configuration in order to determine how to remineralize at the bottom.

I have now described how a model iterating over python dictionaries of tracers and related parameters may provide a structure, which is more manageable for configurations displaying common behaviour such as the ones described in this work, compared to reference designs. With the structure described so far in this section, the model loses some flexibility compared to reference models. There are two remaining issues to solve: 1. Handling tracers showing different

behaviour in the overall logical structure such as having different recycling rates or primary production. And 2. Avoiding having to check for the current active configuration, which is also an issue in other current models. My solution to solving problem 1 is to define classes for the model tracers. The classes then define attributes and methods for the tracer who's functionality they represent. In section 8 I covered the implementation details and expand upon the reasoning for each design choice. The main points were: 1. It becomes possible to define varying behaviour within the framework of pre-calculating values for mortality, primary production etc. which are added to calculate the final contribution to each tracer. 2. Classes are extendible, which allows contributers to easily add functionality without having to copy or rewrite existing code.

With the addition of classes to describe tracers as objects, the biogeochemistry module in Veros has become as flexible as comparable models while being easier to maintain. There is still one issue to with the design as described: It is necessary to check the current active configuration. While that is not a big issue for the two options described in this thesis, a basic NPZD setup with or without a basic carbon cycle. In fact it would be possible to only have a single check for the active configuration. To see why it is desirable not having to know the active configuration, consider a model which not only supports the two configurations described in this work, but also a nitrogen cycle, a calcium cycle, oxygen and iron as well as isotopes. Those are requirements for similar, larger models such as UVic ESCM. Some users may want a certain small set of tracers in order to save computation time, while others need the full capabilities of the model.

If it is necessary to check whether an options is set or not to chose the correct terms of an equation, and these terms themselves depend on other setting of other options, it is a combinatorial problem of creating the correct equations with the correct terms. The more options, the model has, the more possible combinations may influence the equations describing the model. If this is implemented using condition statements, the model may become difficult to maintain, as it lends itself to duplicate definitions and requires outside contributers to know, where to extend or even understand the logical structure.

With an example from UVic ESCM I may demonstrate why I find it important to not rely on checking the active configuration. On the left hand side of Figure 6 is a screenshot of part of npzd_src.F. The font size was reduced to 2px before taking the screenshot in order to fit the code sections where the biodetr varible is updated. The locations where the update happens are highlighted in yellow. The code itself is not intended to be readable. On the right hand side is an exert of the same code in a readable size. Note that in order to update the variable biodetr, it was deemed necessary to check the selected options in order to see, which terms should be added. As can be seen from the yellow markings, it was decided to perform the update in several locations. The updates do not build on top of each other. It is expected that only one of the updates occur for any configuration. It may be seen from the code snippet, that the equations are quite similar with only a few modifications. Such a structure is difficult to maintain, because every edit would have to be done in each of the marked locations. Thus a contributer would have to know all the possible configurations and perform their development across all of them.

My solution to the problem of checking for active configurations is to define rules for the interactions between tracers. The rules return an increase or decrease in one tracer and corresponding increase or decrease in another tracer. I described the strategy and implementation details in section 6. Rules allow building dynamics, which extend each other, which means that a collection of rules describe the full dynamics of the system. If another set of rules were chosen, they would describe different dynamics. A rule is a building block of the model and it is the

```
# if defined O_kk_ballast
    ...
       biodetr = biodetr + dtbio*((1.-bapr)*((1.-dfr)*morp +
           sf + morz)
     & - remi - graz_Det - expo + impo
     & + (1.-bapr)*morp_D*(redntp/diazntp)
# if defined O_npzd_caco3
     & + (1.-bapr)*(1.-dfr)*morp_C
# endif
     & + remi_B)
# else
       biodetr = biodetr + dtbio*((1.-dfr)*morp + sf + morz -
           remi
     & - graz_Det - expo + impo + morp_D*(redntp/diazntp)
# if defined O_npzd_caco3
     & + (1.-dfr)*morp_C
# endif
     & )
# endif
```

Figure 6: Left: Locations highlighted in yellow in npzd_src.F from UVic ESCM where the variable representing detritus, biodetr, is updated. Right: Code example from npzd_src.F showing almost identical equation sets to update biodetr being separated by configuration checks.

collection of selected rules, which specify the system behaviour. The active rules are stored in a list and iterated upon. This allows for having no checks for the active configuration in the logic structure. Therefore, there is only one location in which tracers are updated and when adding functionality, the developer should only be concerned with the function being developed. If a completely different functionality is desired, which cannot be modeled by extending existing classes or adjusting parameters, the user can define a new function with the desired functionality and add it to the selected rules and optionally not select any rules, which it replaces. Since rules contain functions, it is possible to use the defined functions in multiple rules or create templates to be used in several rules. By describing biogeochemistry with rules rather than hard coding functionality, Veros gains great flexibility while being easy to use, read and extend as presented in section 6.

Table 1: Comparison of advantages and disadvantages of previous implementations

Idea	**Advantages**	**Disadvantages**
Reference	Full source-minus-sink equations are visible in code	Duplicate code. Large number of configuration checks. Hard to read.
Dictionaries only	Compact, readable code. Easy to add new tracers. Full source-minus-sink visible	Inflexible, adding functionality outside the assumed behaviour may result in many configuration checks making it hard to read.
Classes and dictionaries	Same as above, but with variability stored within the tracer object. Can implement methods differently. Possibility of storing metadata on objects	May still require configuration check. Dynamics limited to fixed framework.
Rules, classes and dictionaries	Full flexibility, easy to extend. Interaction graphs. Every rule and object available for diagnostics	Must refer to diagnostics to see full source-minus-sink equation

9.1 Flexibility in model extension

The need for the flexibility in biogeochemistry may not appear obvious based on sections 2 and 4, and I have thus far only provided general ideas as to why it would be needed. In this section, I will describe possible extension to the biogeochemistry module and suggestions as to how to implement them.

The final design of the biogeochemistry module was based on the need for a flexible, extensible model, which became apparent, when I implemented Calcium and Nitrogen cycles. Those additional model configurations were not tested with the updated model design, as they were not necessary to conduct the experiment in section 13.

$CaCO_3$ is described implicitly by a smoothing rule in the current configuration. That is $CaCO_3$ itself is not explicitly tracked. It serves as a placeholder to store the calcite produced by phytoplankton and zooplankton and at the end of the time step, the contents are added up and smoothly distributed according to equation (21). That works for my purposes as the effect of calcite on the system. For more accurate simulations it may be required add to know the concentrations of $CaCO_3$ in any cell and redefine the dissolution and remineralization of calcite to work in individual cells rather than as a smoothing rule. In order to do that, it would be required to add a sinking speed to calcite and add an additional type of phytoplankton, coccolithophores, which replaces other phytoplankton in the production of calcite during death. Generally, coccolithophores behave similar to the already implemented phytoplankton. To add coccolithophores it would therefore be sufficient to instantiate an object of the phytoplankton class with appropriate parameters for mortality rates etc. It would be sufficient to reuse the

existing rules for phytoplankton, adjust parameters and register them with a new name. The rules for $CaCO_3$ production and dissolution would have to be modified, as the smoothed distribution no longer applies. The existing smoothing rule would have to be removed from the selected rule set and replaced by rules for dissolution and production. The bulk of the work in adding coccolithophore functionality has already been done, only the addition of rules for production and dissolution would have to be created.

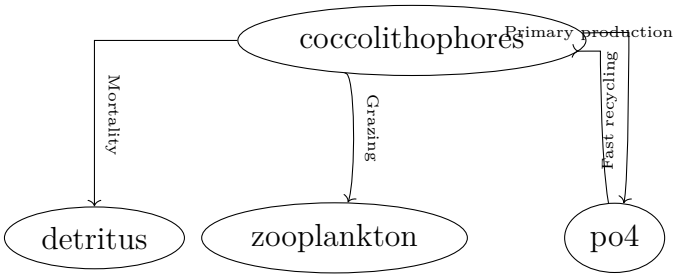


Figure 7: Coccolithophores behave in many ways like ordinary phytoplankton. Adding the plankton type to Veros requires adding an instance of the phytoplankton class and creating new rules adapted for use with coccolithophores.

The addition of a Nitrogen cycle introduces one additional type of plankton, diazotrophs, and 3 additional nutrient tracers, NO_3, DON dissolved organic nitrogen, DOP dissolved organic phosphate. Again diazotrophs are mostly similar to the already implemented phytoplankton. Unlike coccolithophores it is needed to extend the Plankton class. It would be required to create a new method for primary production, which limits growth compared to phytoplankton. It would also be necessary to replace existing rules, as consumption of PO_4 should now only occur, when it is present in higher quantities than DOP. Had I had to implement such functionality without the rule model, I would have had to include conditions every time a calculation would involve the phytoplankton tracer or DOP. Rules should be added for diazotrophs similar to those of phytoplankton and additional added for the consumption of NO_3 and DON. The nitrogen cycle (figure 27 in Appendix A) as implemented in UVic ESCM, also adds bottom denitrification, which requires adding a post rule smoothing results from previous calculation.

Each of these additions are possible to add on top of the basic NPZD setup. By facilitating the rule model, I have made it possible to easily extend the functionality with a new plankton type as in the $CaCO_3$ cycle. It is possible to modify behaviour as shown with the Nitrogen cycle by creating new rules for the modified behaviour and selecting them, rather than the rules, they replace.

The addition of each of these extensions will result in extended dynamics, which are shown with graphs auto-generated by the diagnostics module in Appendix A. Each of the configurations can be run with Veros without checking for which rules or tracers are active.

It should be noted, that these examples are not added to the current Veros release. They serve as examples of extension and the need for a flexible model setup.

10 Diagnostics

Combining equations (2) to (5) and following the description in section 6 it can be seen, that since every change to a tracer in the basic model of section 2 is balanced by a corresponding change in another tracer. That is, the NPZD dynamics as described in this thesis is a closed system. Therefore it is possible to define a conserved measure. I chose the total phosphorus as the conserved property, because it is present in either of the described configurations and is fully described by the dynamics:

$$P_{total} = \iiint (R_{P:N}P + R_{P:N}Z + R_{P:N}D + \mathrm{PO}_4)dxdydz \tag{83}$$

The conserved property may be monitored from the diagnostics module. I could have defined it in terms of any of the other tracers, and should the model be extended with additional tracers, the conserved measure would have to be updated.

$$P_{total} = \iiint \left(\sum_i f_i T_i\right) dxdydz \tag{84}$$

With T_i, and f_i the tracer and phosphate concentration factor for the tracers in the model. It may be convenient during development to also monitor total carbon in a similar fashion, although it is not conserved when including carbon exchange with the atmosphere, since there is currently no atmosphere component in Veros and the atmospheric CO_2 concentration is assumed constant.

One thing to be aware of when monitoring total PO_4 is, that the biogeochemistry module asserts a minimum tracer concentration. This assures, the model never carries negative tracer values, but when each rule is evaluated the contributions to each tracer is added up, and the total reduction in a tracer may rise beyond the present concentration in any cell. During the main biogeochemistry loop, every tracer concentration is checked after each evaluation of the loop. When a tracer concentration drops below zero, a flag is set marking the cell as depleted. The concentration in the cell is then set to a minimum value and the cell is excluded from calculations which consumes it causing reductions during the remaining loop iterations for the time step. By excluding the depleted cells from calculations the cells may maintain a minimum concentration without disturbing the total amount of phosphate. The only time total phosphate can be changed is the bio time step when a tracer concentration in a cell is brought below 0. By keeping the time step size low, such drops below zero concentration may be reduced at the cost of evaluation time. I have found, that bio step size below 6 hours maintains concentration in a calculation scheme without physical transport within a grid of 4 degrees separation. The transport does not guarantee exact conservation of total phosphate. (Kvale et al., 2015; A. Schmittner, Gruber, et al., 2013) which implements the same dynamics recommend a time step shorter than 3 hours. UVic ESCM does not conserve tracers exactly during transport either (A. Schmittner, A. Oschlies, et al., 2005). If a model setup requires adding phosphate forcing, that would have to be accounted for in equation (84).

Veros supports the notion of diagnostics modules. That is a class who's diagnose method is called at a user determined interval. The intention of the diagnostics is to provide information during model evaluation to the user as well as saving data for later inspection. (Häfner, Jacobsen, Nuterman, et al., 2018) There are built-in options for snapshotting values and averaging which are both useful for the biogeochemistry as well as diagnostics for overturning and energy conservation. I have created a diagnostics class specifically for the biogeochemistry functionality. This class is responsible for creating the interaction graphs shown in this work. The graphs

are created by displaying the registered tracers as nodes with the name specified during registration. Each of the active rules are drawn as edges between the source and the sink, which were registered during rule creation. Each edge is labelled with a label which can optionally be set when creating the rule. The edges display whether the rule is a PRE rule, a PRIMARY rule or a POST rule. PRE rules are displayed with dotted arrows, PRIMARY rules with solid arrows and POST rules with dashed arrows.

The diagnostics class also allows for monitoring total phosphorus as described by equation (84). Since the change in total phosphorus should remain 0 in a closed model, like the one described, it is useful to enable this value during development of additional rules. If further tracers are added to the model,the diagnostics should be updated to reflect that change.

The diagnostic also provides the option for printing results of calculated values used in rules such as primary production, excretion, mortality, recycling, etc. These can be printed for any desired tracer. Having such an option provides users with the ability to monitor individual terms in the source-minus-sink equations for a specific set of defined tracers, which may be helpful during development of added functionality. If so desired, it is also an option to save snapshots of these values. Finally the biogeochemistry diagnostics allow evaluation of a selection of rules allowing their output to be monitored. This acts as an extension of the ability to individually select rules. By monitoring individual rules, it is possible to monitor their effects on a larger, interconnected model setup.

11 Employing a user kernel

The CO_2 flux calculations use a Newton-Raphson solver in every surface grid cell. When using a solution in pure Numpy it would be required to perform the calculations in every cell until all cells had reached a solution or exceeded the maximum allowed iterations. Should a few cells require more iterations than the bulk, it would cause a large amount of wasted calculations. Furthermore attempting to calculate such a flux over land would be meaningless.

It is possible to reduce the amount of cells active in the calculations by applying a mask to the grid indicating which cells are done calculating starting by masking out any land cells. As the calculations progress any cell with a sufficiently accurate solution would be marked complete and excluded from further calculations. A potential problem with this approach is, that masking arrays is handled by Numpy, which means that when using the Bohrium backend for Veros, the data set would require syncing from Bohrium to Numpy, which is an expensive operation. Numpy performs the masking single threaded, which means there is a potentially large performance loss as the model grows. A solution to the problem is to write a user kernel for Bohrium, which is aware of the mask and can skip the masked cell without syncing to Numpy. A masked, bounded Newton-Raphson solver could be useful for others. Therefore I decided to write such a user kernel and compare the performance of a pure Numpy solution, a Bohrium accelerated version of the Numpy solution and the custom user kernel. The results are presented in figure 8. When using the Numpy backend, there is no significant difference in performance between using the masked and unmasked solvers. The Bohrium backend however performs worse in the flux calculations in both cases with the masked solver significantly worse than the unmasked. This justifies the need for a user kernel. When indexing Numpy arrays with a mask, Bohrium uses Numpy itself to do the indexing, which means it must copy the data back and forth between Bohrium and Numpy. This is the cause of the decreased performance. I wrote a kernel for OpenMP, which works by skipping cells, which have been marked completed.

This eliminates the need for syncing to Numpy and regains the benefits of using the Bohrium backend. Since the kernel was written for OpenMP, there is no additional performance gain from the kernel when using Cuda or OpenCL with Bohrium unlike the remaining parts of the code, which can be automatically parallelized which can also be seen in the figure.

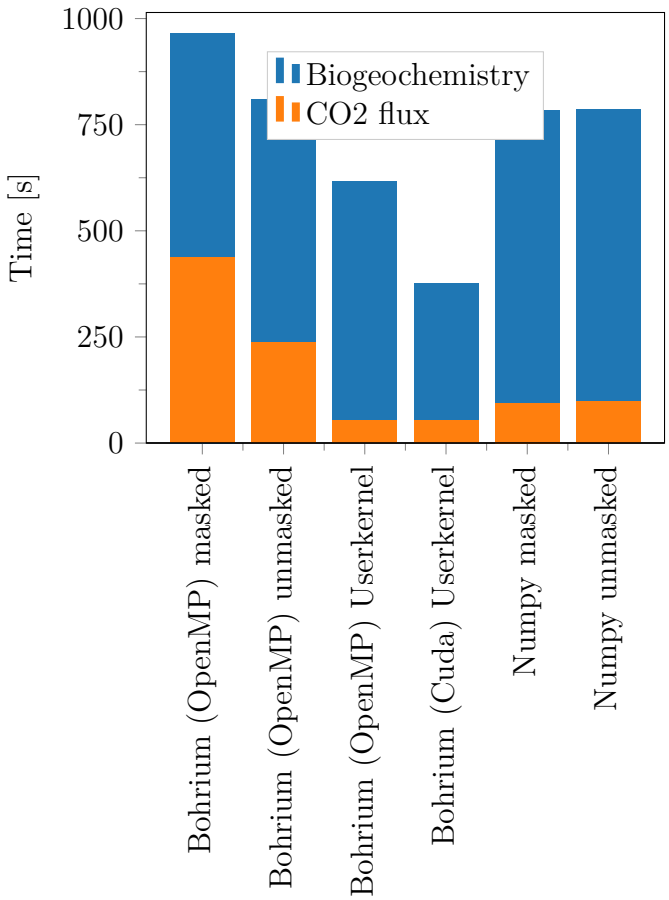


Figure 8: Benchmarks of the biogeochemistry module on a $128 \times 64 \times 45$ grid for 2880 time steps average of 3 runs. Lower is better. The machine in use had an Intel Xeon E5-2650 CPU with 12 cores @ 2.2 GHz and an NVIDIA P100 GPU.

Implementation of the flux calculations in Veros is based on a reference implementation in UVic ESCM (Andreas Schmittner, 2018) which bases its implementation on the rtsafe method from Numerical Recipes (Press, 2002). Rtsafe is a Newton-Raphson solver with boundaries, which performs a bisection step rather than the Newton-Raphson step if the Newton-Raphson step would take the solution out of bounds. The recommendation listed in the UVic ESCM code (Andreas Schmittner, 2018) is to initially set the boundaries at oceanographic safe values of $[H^+]_{max} = 10^{-6}$, $[H^+]_{min} = 10^{-10}$ and reuse the result of the previous run to set the boundaries for the next run corresponding to pH $\pm$ 0.5. With the pH values defined as $-\log_{10}[H^+]$. The initial guess for a solution is placed in the middle between the boundaries. Reusing the previous result as initial guess for the next iteration often aids in fast convergence. Hence the idea of basing the boundaries on the previous result is sound. However the way it is done based on the pH-value shifts the initial guess towards the boundaries, $10^{<x,y>} \neq < 10^x, 10^y >$. To preserve the previous result as initial guess, I use a slightly different approach. I set the boundary width to be twice the shortest distance from the previous result to either safe boundary but at minimum

1/5 of the distance between the safe boundaries. The actual minimum boundary width does not affect the convergence rate significantly. Its purpose is to prevent a cell getting stuck at one of the boundaries with too small boundary distance causing it to converge on an incorrect value. By reusing the previous result as initial guess, a large part of the cells will already be at a sufficient degree of accuracy before the first iteration. On a 128×64 grid with time step size 3300 s half the cells completed the rtsafe routine in the first iteration. The following iterations had less than 1500 cells remaining, which usually completed within the second iteration. When using boundaries based on pH-value the average required number of iterations increases to 6 with most cells active until the final iteration.

12 Model evaluation

In order to evaluate the validity of the implemented model I compare the output of a 400 year simulation with measurements from the Global Data Analysis Project, GLODAP, data set (Key et al., 2004) for DIC and Alkalinity, and data sets from the Sea-Viewing Wide Field-of-view Sensor project, SeaWIFS, (Group, 2015) for phosphate and phytoplankton.

The results shown in this section are from a coarse resolution simulation with a $90 \times 40 \times 15$ points grid. Because of the coarse resolution it does have some issues resolving certain aspects of the global circulation, which affects the biogeochemical system. For example with this resolution upwelling on the pacific equator is weak. The chosen model parameters were selected to compensate for that. The minimum vertical diffusivity, $\kappa_H^{\min}$, was set to $7 \times 10^{-5}\,\mathrm{m^2\,s^{-1}}$ and the detritus sinking speed was set at $2\,\mathrm{m\,d^{-1}}$. The remaining model parameters were selected to fit measurements as accurate as possible.

Table 2: Model parameters

Parameter	**Symbol**	**Value**	**Units**
Light attenuation through phytoplankton	k_c	0.047	$\mathrm{m^{-1}/(mmol^3)^{-1}}$
Light attenuation through water	k_w	0.04	$\mathrm{m^{-1}}$
Maximum growth rate	a	0.23	$\mathrm{d^{-1}}$
Specific mortality rate	μ_P	0.035	$\mathrm{d^{-1}}$
Fast recycling rate (microbial loop)	μ_{Pt}	0.025	$\mathrm{d^{-1}}$
Half saturation constant for N uptake	k_N	0.7	$\mathrm{mmol\,m^{-3}}$
Assimilation efficiency	γ_1	0.5	
Maximum grazing rate at 20 °C	g	0.13	$\mathrm{d^{-1}}$
Growth efficiency	ge_Z	0.6	
Mortality	ν_Z	0.06	$\mathrm{mol^{-2}\,d^{-1}}$
Detritus remineralization rate	ν_D	0.09	$\mathrm{(mmol/m^3)^{-2}/d}$
Detritus base sinking speed	wd_0	2	$\mathrm{m\,d^{-1}}$
Detritus sinking speed increase with depth	mw	0.02	$\mathrm{m\,d^{-1}\,m^{-1}}$
Detritus maximum depth for speed increase	mw_z	1000	m
Calcite dissolution depth	d_{CaCO3}	1500	m
c	c	1	$\mathrm{°C^{-1}}$
b	b	1.038	
Carbonate to carbon production ratio	Ca_{pr}	0.022	

Parameters set for the simulation are presented in table 2. For initial conditions I used a uniform distribution of nutrients and detritus with concentrations for PO_4 of $2.5\,\mathrm{mmol\,m^{-3}}$, $2300\,\mathrm{mmol\,m^{-3}}$ for DIC and $2400\,\mathrm{mmol\,m^{-3}}$ for Alkalinity with the concentration of detritus at $1 \times 10^{-4}\,\mathrm{mmol\,m^{-3}}$. Surface concentration of phytoplankton was initially set at $0.14\,\mathrm{mmol\,m^{-3}}$ and $0.014\,\mathrm{mmol\,m^{-3}}$ for zooplankton both with an e-folding depth of 100 m. External input to the model for heat fluxes, freshwater fluxes and shortwave radiation are using climatological monthly mean values from the CORE.2 Global Air-Sea Flux Dataset (*CORE.2 Global Air-Sea Flux Dataset* 2008).

12.1 Results

The geographical distribution of phytoplankton is determined by the availability of nutrients and light. Therefore the model should produce the largest populations in areas with upwelling near the surface: The Southern ocean and northern parts of the Pacific and Atlantic oceans as well as in the Eastern equatorial pacific. Detritus and zooplankton should be present at the same locations, because zooplankton grazes on phytoplankton, and detritus is produced in areas with phytoplankton and zooplankton. As detritus sinks, it will have a presence at deeper levels and zooplankton will follow. The SeaWIFS project provides annual means of measurements of chlorophyll A concentration from satellite measurements. This data is shown in figure 9 along with modelled concentrations. The data set contains large variability between measured points. Therefore the color scheme was chosen such, that the figure displays distinct spacial features. In particular I wanted to be able to display equatorial concentrations. This means that in certain areas, the concentration reaches the maximum color scale. For the modelled concentration I selected a slightly different color scale, which would display the modeled overestimation of equatorial plankton concentration. The plankton concentration in $\mathrm{mmol\,m^{-3}}$ is converted to mg chlorophyll $\mathrm{m^{-3}}$ by multiplying the plankton concentration by $1.59\,\mathrm{mg\,mmol^{-1}}$ (A. Schmittner, A. Oschlies, et al., 2005). The plankton concentration is taken as the mean concentration of the upper 850 m, as the phytoplankton population is confined to this area as shown in figure 13.

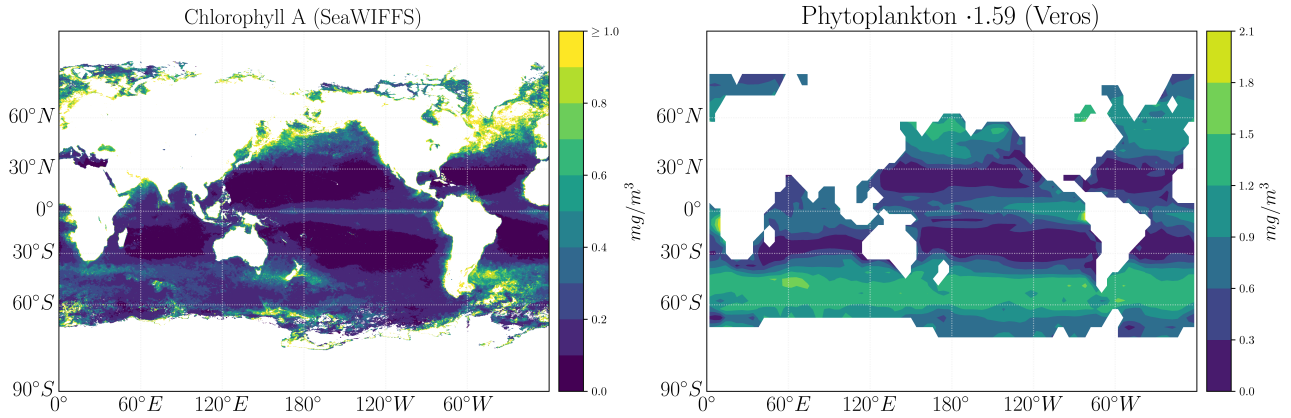


Figure 9: Left: Chlorophyll a concentration as measured by SeaWIFFS. Right: Veros model output from a 4deg resolution model for phytoplankton converted to chlorophyll A

The SeaWIFS measurements show an increased chlorophyll concentration in areas with upwelling north of 35 °N and south of 35 °S. This is also evident in the simulated data. The measurements also show a population along the equator in both the Pacific Ocean and the Atlantic Ocean. I have been able to reproduce the presence at the equator, but the simulated concentration is too high. The model parameter set was chosen in such a way, that there would be a phytoplankton surface population at the equator. This parameter choice should compensate for the difficulty for the low resolution simulation to produce upwelling along the equator. Chlorophyll measurements from the SeaWIFS project are based on images of the surface and comparing a mean concentration of the upper 850 m may be a too large depth. As the plankton distribution is largest near the surface, integrating over lower depths provide larger mean concentrations, which could indicate the phytoplankton concentration is even further exaggerated.

Figure 10 displays vertical integrals of the tracers from the sea surface to 300 m depth divided by the integration depth to provide a mean concentration.

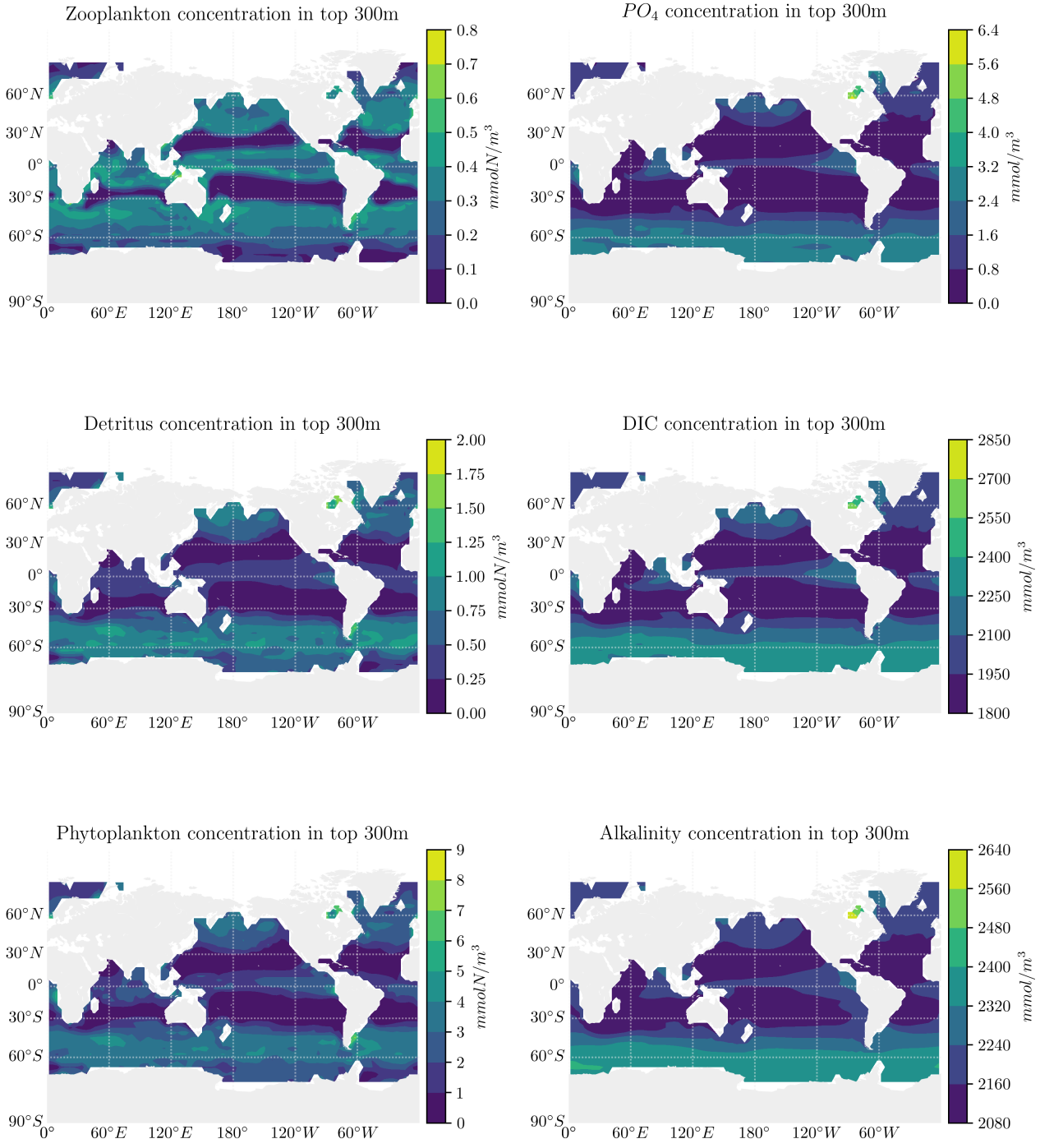


Figure 10: 300m vertical integrals of yearly averaged concentrations for Veros tracers

Looking at both the surface level in figure 11 and 300 m concentrations the geographical distribution of plankton is similar to the vertical integrals, as the highest concentrations are at the surface. In the same figure it may also be seen that the phytoplankton growth is confined to areas with phosphate near the surface. This phosphate distribution is consistent with the measured surface concentration shown on the right hand side in figure 12.

It may also be useful to consider the mean concentration of tracers at vertical levels as shown in figure 13. The zooplankton and phytoplankton distributions are confined to the surface layers, which is also reflected in the phosphate concentration. Phosphate is being consumed in regions with phytoplankton and remains at a concentration around $2.4\,\mathrm{mmol\,m^{-3}}$ at depth. DIC is consumed by phytoplankton and produced by remineralization. Further amounts of DIC are removed from the surface and redistributed towards the bottom by the formation and dissolution of calcite. There is also a contribution to surface concentration by the exchange of carbon dioxide with the atmosphere. However the increased deposit of CO_2 in some areas will be balanced by release in other areas. As described in section 4 $CaCO_3$ redistributes with a larger amount closer to the bottom causing a net decrease at the surface and net increase at depth. This contribution is twice as large for alkalinity as it is for DIC, which results in a more pronounced gradient for alkalinity. Detritus is produced in areas with plankton and is expected to follow the phytoplankton and zooplankton distributions. As detritus also sinks, the vertical distribution is dragged downwards

Similar conclusion may be drawn from the zonal averages presented in figure 14, which furthermore shows the increased nutrient content regions with upwelling and how the plankton distribution follows.

The PO_4 concentration generally matches measurements. The surface distribution as measured by the SeaWIFS project, figure 12, show the largest concentration around 60 °S and in the northern Pacific. This is also present in the simulated data. Additionally there should be a smaller elevated concentration in the eastern equatorial Pacific. With the chosen parameters, I was not able to produce that concentration in the surface layer, but it may be seen in lower layers in figure 10

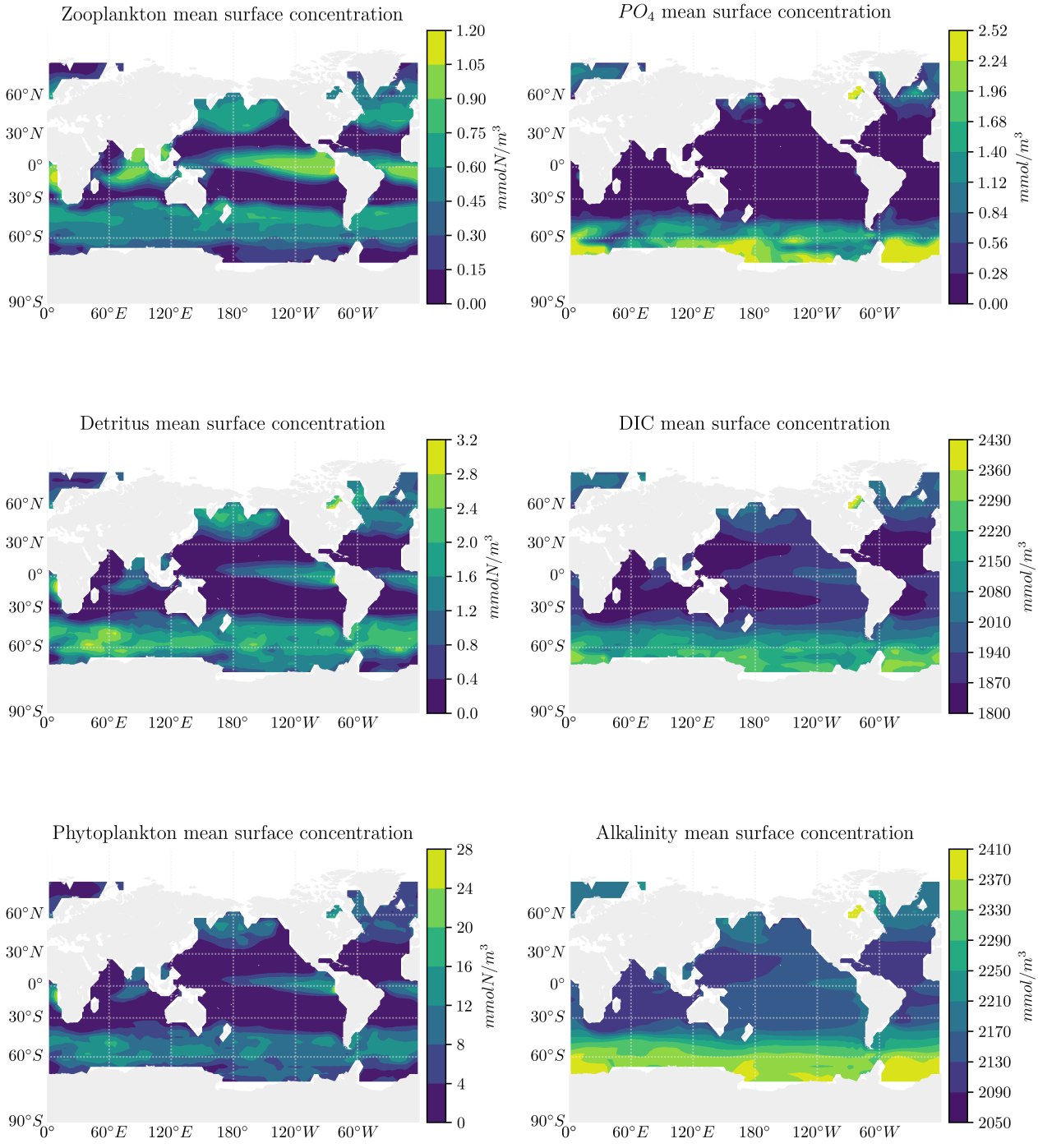


Figure 11: Yearly mean surface concentrations of Veros tracers

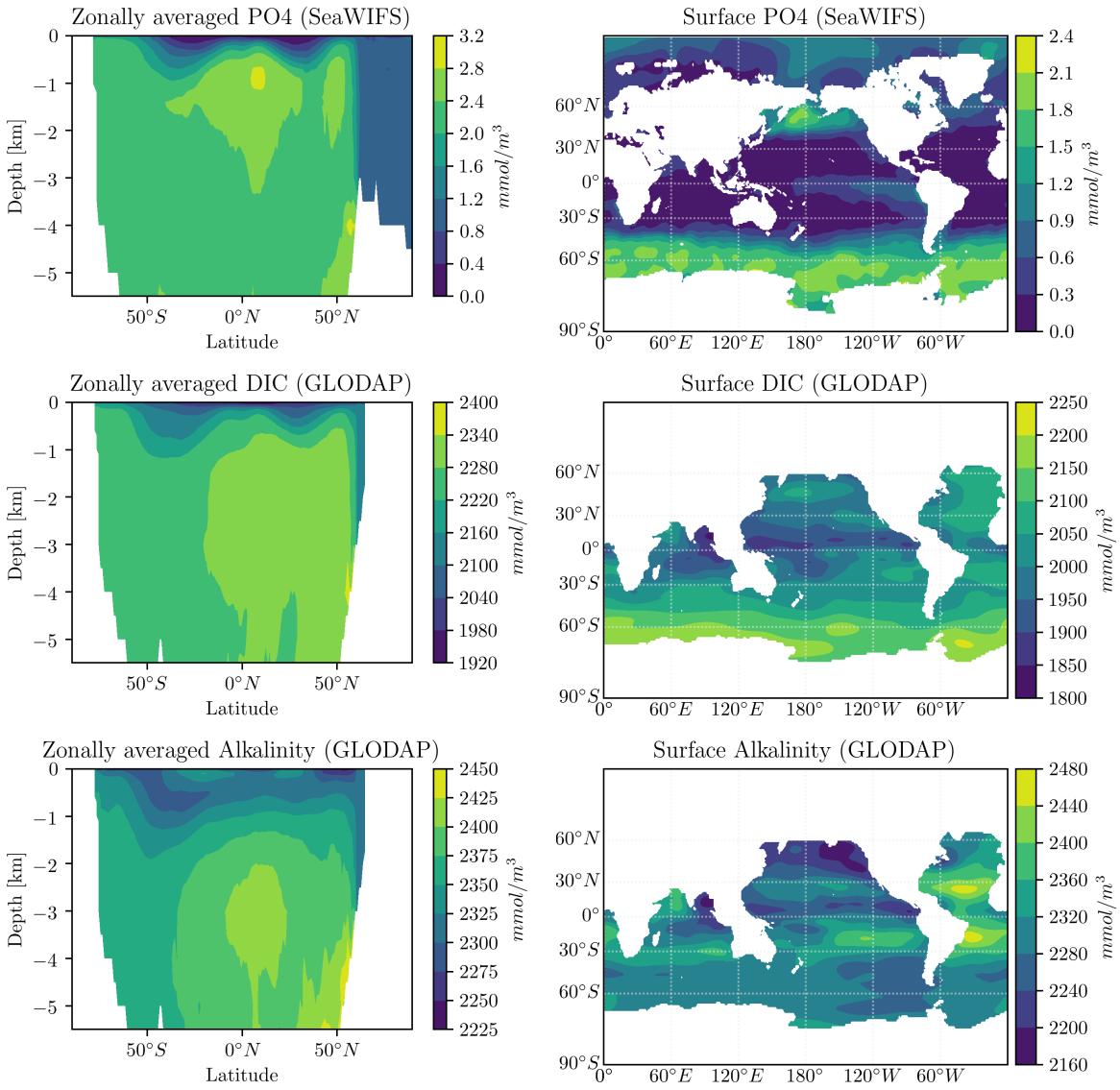


Figure 12: Left: Zonal averages of reference data sets. Right Surface concentrations of reference data.

Phytoplankton concentration is the primary source of growth in zooplankton, and contributes to detritus concentration as well. Therefore its distribution is of importance to the distribution of the other tracers.

The phytoplankton distribution follows expectations in that it is present in larger concentrations at high latitudes and in equatorial regions with upwelling of nutrients. In the intermediate regions between the polar regions and the equator in the Pacific ocean, large gyres remove nutrients and plankton from the surface layers, which limits plankton growth. As zooplankton and detritus are both strongly dependent on phytoplankton to sustain their populations, they are distributed accordingly with slightly deeper presence due to detritus sinking. This distribution again is according to expectation. With the presented parameter configuration, the phytoplankton concentration is generally too high in particular at the equator, but matches expectations in spacial distribution. The increased population also causes an increase in zooplankton and detritus concentrations. Since phyto- and zooplankton mortality are the sole modelled drivers

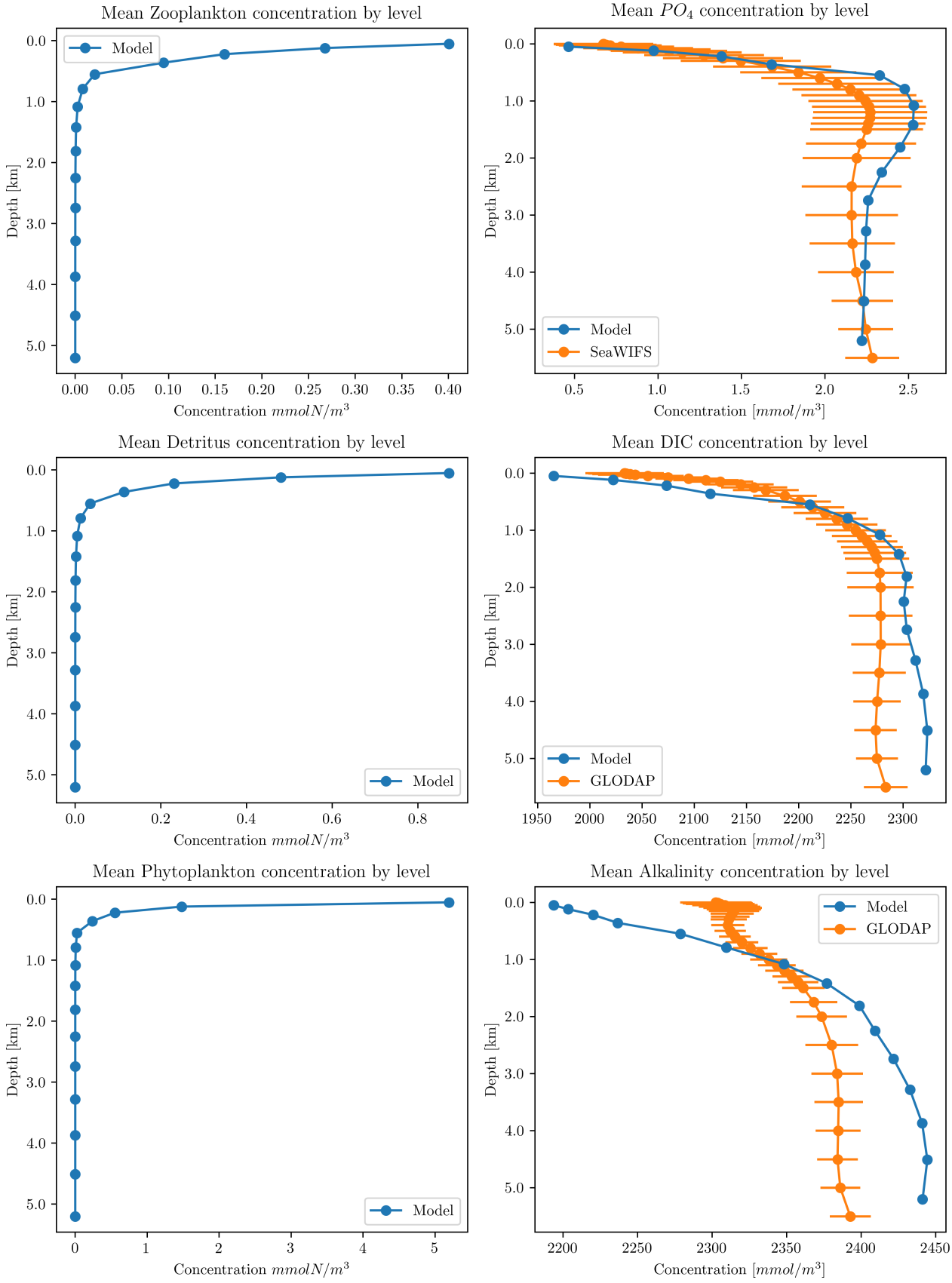


Figure 13: Mean concentrations of model tracers at each vertical level along with GLODAP and SeaWIFS measurements where available.

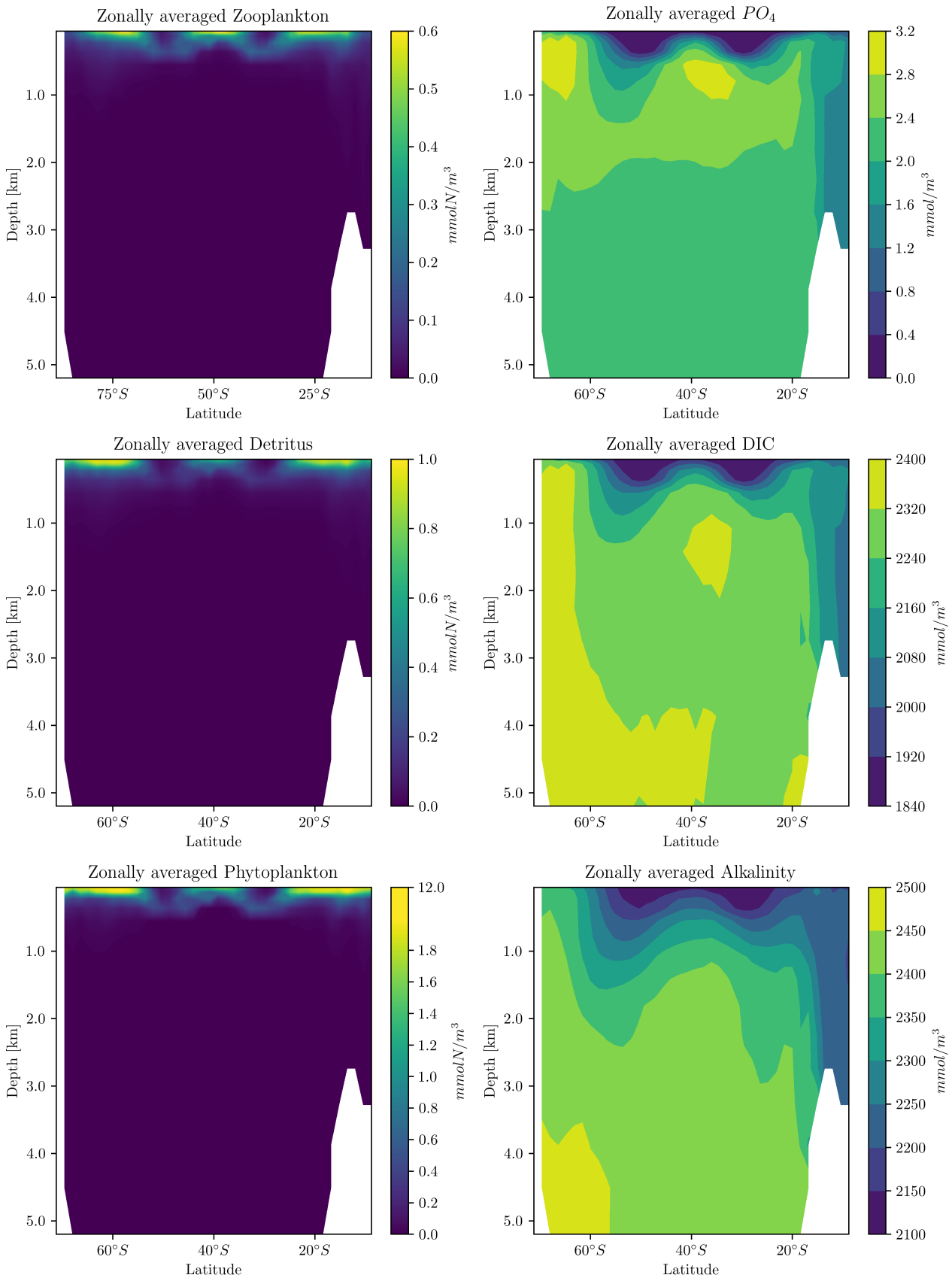


Figure 14: Zonally averaged tracer concentrations

of $CaCO_3$ redistributions, which influences DIC and Alkalinity by removing additional material at the surface and adding it back at depth, a too large surface plankton population causes a reduction in surface concentration for those tracers as evident in figure 13.

Figure 14 presents zonal averages of tracer profiles. As may be expected the plankton tracers are primarily distributed towards the surface where there is light. As for PO_4, DIC and Alkalinity they all display higher concentrations in areas with upwelling in regions north of 50 °N and south of 50 °S as well as in the equatorial region between 10 °S and 10 °N From the same plots, a reduced concentration in the shallower region north of 70 °N which is also present in the GLODAP measurements. For PO_4 the zonal average follows the SeaWIFS data set.

The average concentration in each of the model layers are depicted in figure 13. Although the zonal average for PO_4 is reasonable, the vertical profile shows an overestimated concentration at 1000 m depth and a decreasing concentration above.

The low surface phosphate concentration along the equator suggests, that the phytoplankton population is reliant on replenishing of the phosphate concentration from below, which was the purpose of increasing the minimum vertical diffusivity, κ_H^{min}, at $7 \times 10^{-5}\,\mathrm{m^2\,s^{-1}}$. This was chosen over the default value for the 4deg Veros model of $\kappa_H^{min} = 2 \times 10^{-5}\,\mathrm{m^2\,s^{-1}}$, because the model could not accurately replicate upwelling in the equatorial region with lower minimum vertical diffusivity. The too large nutrient concentrations in the Southern Ocean may be attributed to the larger diffusivity.

The sinking speed of detritus could also be used to regulate the phosphate distribution, as it affects the vertical distribution via remineralization. Modifying parameters for phytoplankton mortality or its fast recycling rate also influence the population size and nutrient distribution.

In figure figure 13 the alkalinity gradient caused by the production and dissolution of calcium carbonate is too large, which as mentioned to an extend was expected by the large plankton concentration. I am not able to reproduce the increased surface alkalinity in the Atlantic Ocean the GLODAP measurements show. This affects the global mean and therefore the part of the deviation from measurements. DIC follows the measurements from GLODAP well, except for a generally increased concentration below the calcite dissolution depth at 1500 m and decreased concentration above. This same pattern of may be seen for Alkalinity.

To fit the vertical DIC and Alkalinity concentrations, it is also possible to adjust the calcium to detritus production ratio, which adjusts, how much calcium is remineralized from dying plankton. Reducing this ratio adjusts the gradient of the vertical profile shown in figure 13 closer to the measurement data. However since the modelled phytoplankton concentration is overestimated and the plankton mortality is responsible for the $CaCO_3$ redistribution, a better focus for increasing the model accuracy would be to reduce the plankton concentration.

Other simulations with different parameter sets gave results with significantly reduced or nearly depleted surface concentrations of both phosphate and phytoplankton (not shown). When running the model with a higher spacial resolution, it is not needed to increase the vertical background diffusivity to recreate upwelling along the equator. This in turn generally reduces the surface nutrient and plankton concentration. Both indicative, that it may be possible to further optimize the parameter set.

Figure 15 shows the geographical regions with CO_2 absorption and release as compared to measurement data. At the top of the figure are filled contour maps of the flux. On the bottom are zonally averaged values. The left hand side of the figure is modelled values from Veros, and on the right are measurement data. The CO_2 flux is generally consistent with the data from (Takahashi et al., 2009). Although there is increased upwelling of DIC in the Southern ocean, which causes an exaggerated flux into the atmosphere. The flux is shown with positive values

indicating a flux from the ocean into the atmosphere.

The zonal averages for DIC and PO_4 indicate, that there is increased nutrient transport to the Southern ocean. This in turn causes an elevated CO_2-flux as well as higher plankton population in the area. Since the simulation was run with an atmospheric CO_2 concentration of 280 ppmv comparable to preindustrial values, and the reference data are from more recent measurements, the simulation can show a higher degree of outgassing than the reference measurements.

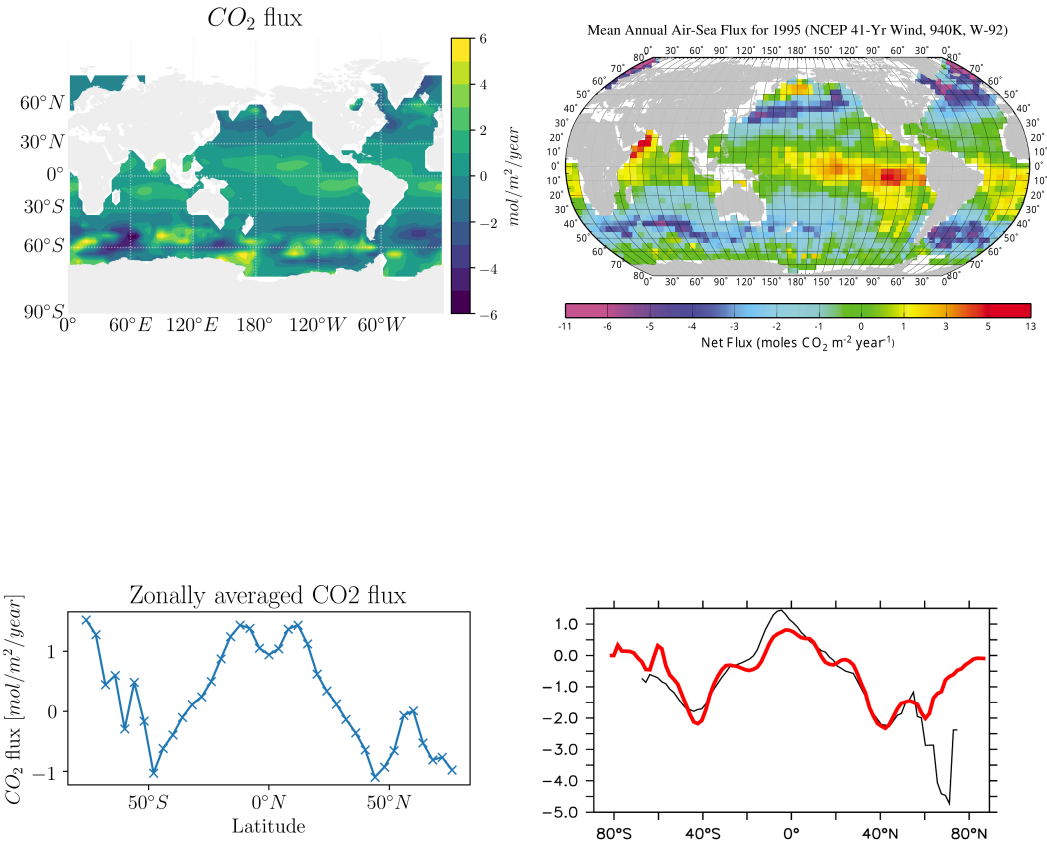


Figure 15: CO_2 flux over the ocean-atmosphere boundary. Left: Model output, Right: Reference data from (Takahashi et al., 2009) Top: Contours of flux, Bottom: Zonally averaged flux. The reference image is from (Andreas Schmittner et al., 2008). The zonally averaged reference data is shown with the black line. The red line is UVic ESCM output.

13 AMOC Collapse

The Atlantic Meridional Overturning Circulation [AMOC] is subject to research due to its influence on global climate. The AMOC asserts large control over heat transport and storage of chemical species, and has been involved in climatic temperature changes of several degrees Celsius over the course of a few decades. (Kuhlbrodt et al., 2007)

In this section I describe an experiment, in which the AMOC is weakened by introducing a freshwater flux in the North Atlantic reducing the salinity. I then analyse the resulting effects on the global biogeochemistry.

The AMOC consists of four parts. Upwelling from depth to near the ocean surface, surface currents transporting light water towards higher latitudes, deep water formation sites where water density increases and sinks, deep ocean currents which close the circulation.(Kuhlbrodt et al., 2007)

Wind and tides generate turbulent mixing. Mixing of heat lightens water masses at depth, allowing them to rise at low latitudes. The resulting surface water mass is then transported northward. Due to atmospheric cooling and ice formation, which rejects salt, the water parcels increase in density causing them to sink and disperse. The result is increased deep water mass and a resulting meridional density gradient between high and low latitudes. The circulation is illustrated in figure 16.

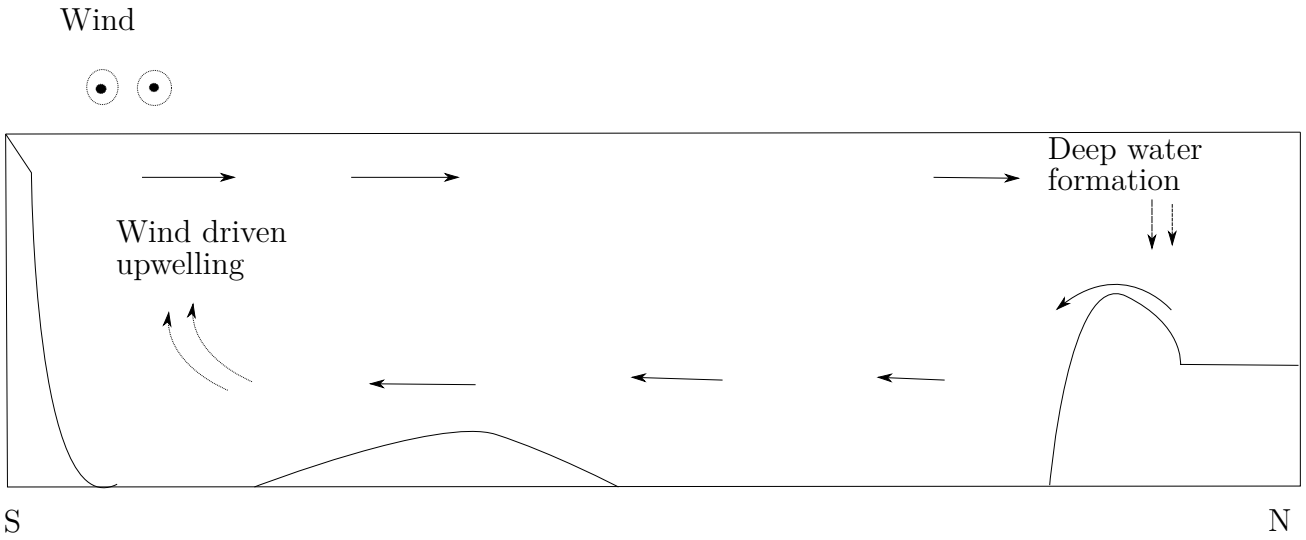


Figure 16: Simple schematic of the AMOC

Adding freshwater fluxes in the northern Atlantic can weaken the circulation by reducing the salinity increase due to freezing water and thereby the density increase, which is responsible for water parcels sinking. Without this part of the circulation the density gradient is reduced and the circulation weakened.

The AMOC carries large amounts of nutrients from deep waters to the surface, which is necessary for plankton survival. Without influx of nutrients to the surface layers, plankton populations would consume the available nutrients until depletion or until a smaller, stable population size is reached. The formed detritus then sinks towards the bottom to be remineralized. When the circulation is weak, deep water nutrients would remain trapped.

13.1 Model setup

The Veros model was first initialized with the same conditions as in section 12. The model then simulates 200 years to reach a state representable of current ocean conditions. With that result as initial conditions, I then simulated additional 200 years as a control experiment. With the results of the first 200 years as initial conditions the experiment was conducted again this time including a freshwater flux north of 50N and between -100W and 50E by reducing the sea surface salinity by 1 PSU in the salinity forcing calculations. The area is shown in figure 17 It should be mentioned, that I only changed the salinity in the affected region and not any of the other model tracers, although these would also have been affected by a freshwater flux. I intended only to weaken the AMOC without modifying other parts of the model, therefore changing the salinity was sufficient.

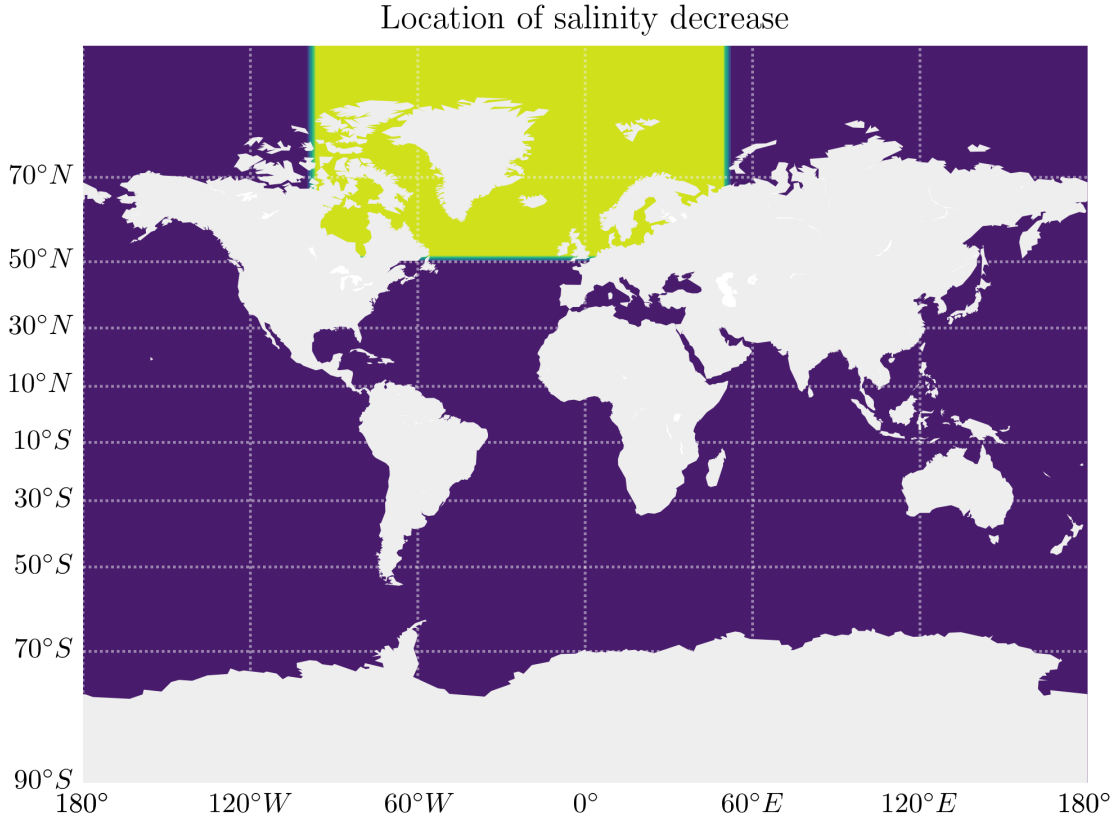


Figure 17: In the yellow area the sea surface salinity was modified to cause a collapse of the AMOC

13.2 Results

After introducing the freshwater flux, the deep water southward transport should be limited in the weakened circulation with one of the drivers limited. This is evident in figure 18. With a weakened AMOC the surface currents carrying light warmer water to higher latitudes are

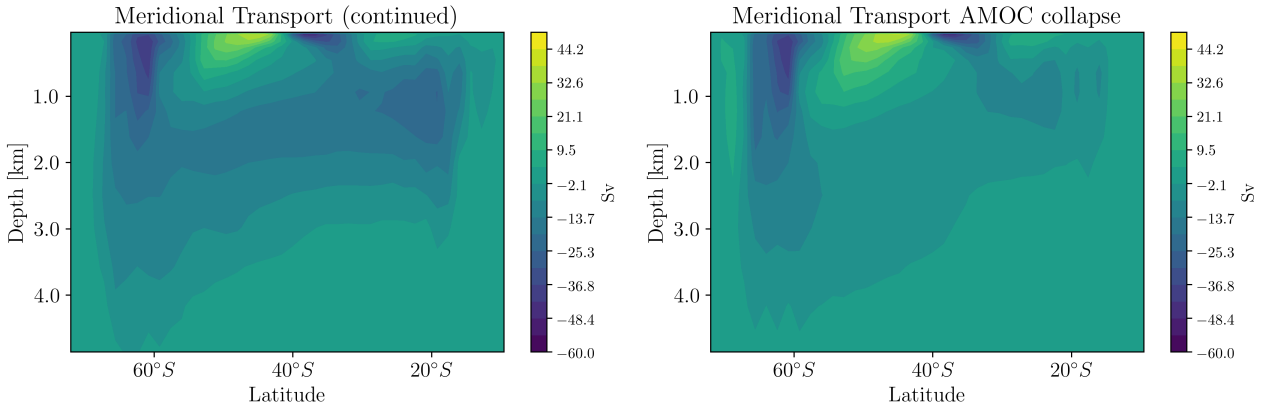


Figure 18: Meridional transport for the control experiment (left) and the AMOC collapse experiment (right). The southward transport below 1km is reduced when collapsing the AMOC.

also weakened and have limited ability to transport heat. This causes a reduction in surface temperature in the North Atlantic as seen in figure 19. This reduced heat transport from the Southern to the Northern hemisphere is often seen in simulations which collapse the AMOC (Brown and Galbraith, 2016). With reduced transport is typically found an increased surface temperature around 30 °S. I found a similar temperature anomaly in the Southern Atlantic in figure 19.

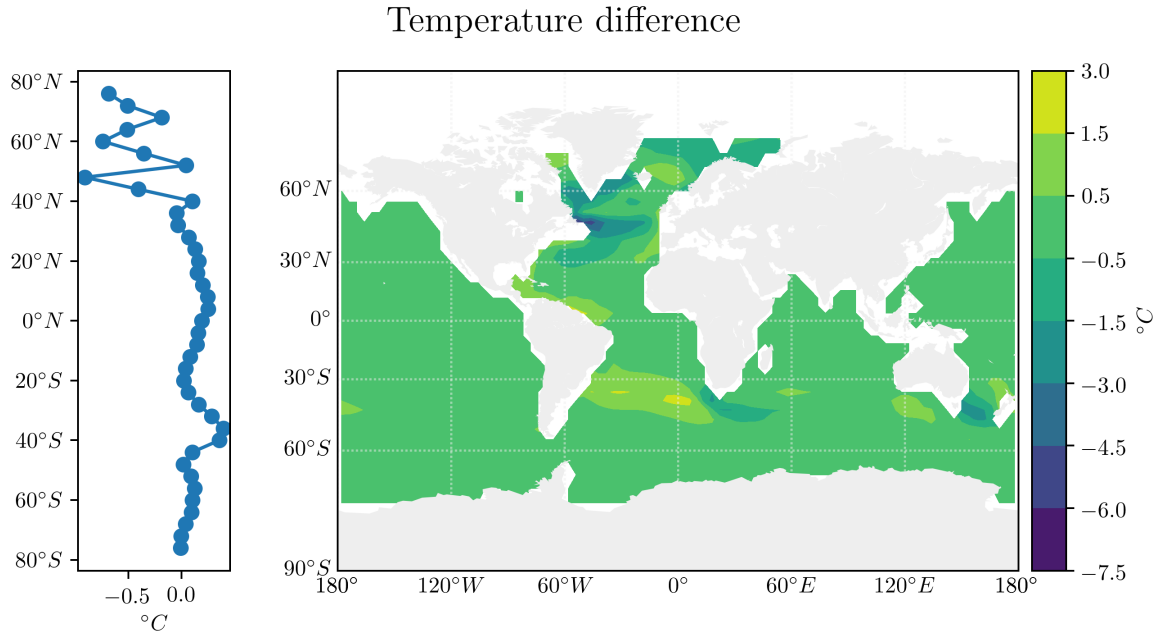


Figure 19: Mean difference in surface temperature in the upper 220 m between the reduced AMOC simulation and control experiment after 200 years of simulation. The sea surface temperature in the North Atlantic is significantly reduced. Left: Mean difference by latitude

With lower temperature and limited nutrient resupply the plankton concentration subsequently reduced as shown for phytoplankton in figure 20. As the formed detritus sinks before being remineralized and there is no overturning to bring the nutrients back up, the nutrients are effectively moved from the surface to depths further limiting plankton growth. This is con-

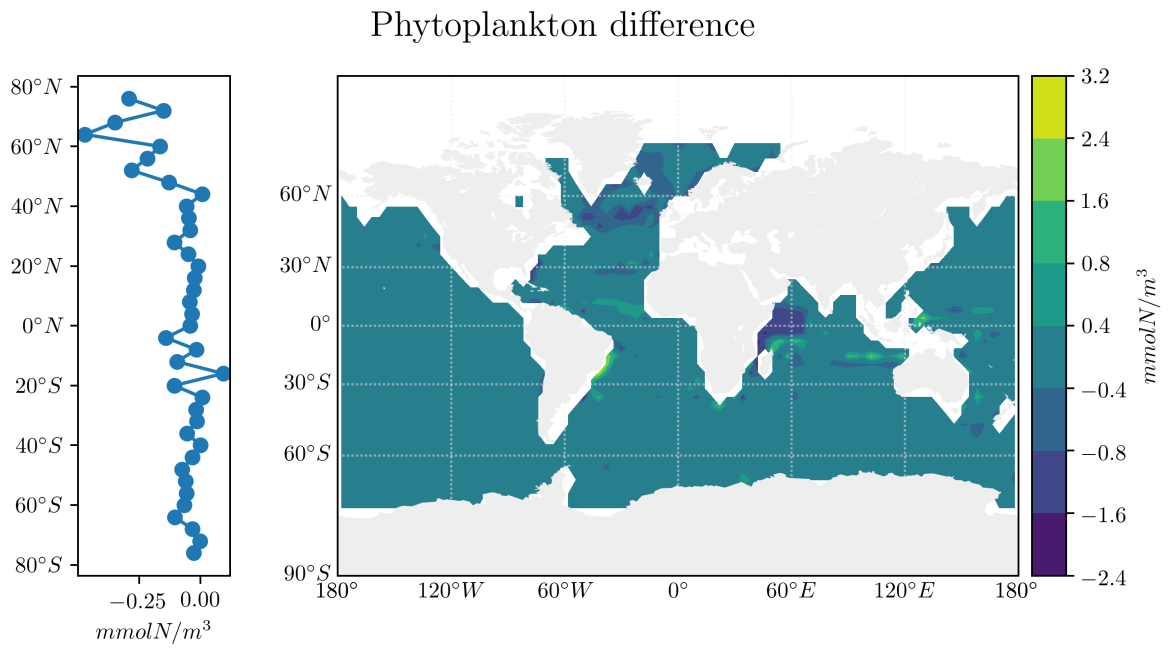


Figure 20: The mean phytoplankton concentration in the upper 220 m is reduced after the weakening of the AMOC due to a reduction in temperature and nutrients. Left: Mean difference in concentration by latitude.

sistent with similar experiments (Nielsen et al., 2019). The reduced surface DIC is shown in figure 21, and a zonal average of difference between the experiment and control is shown in figure 22.

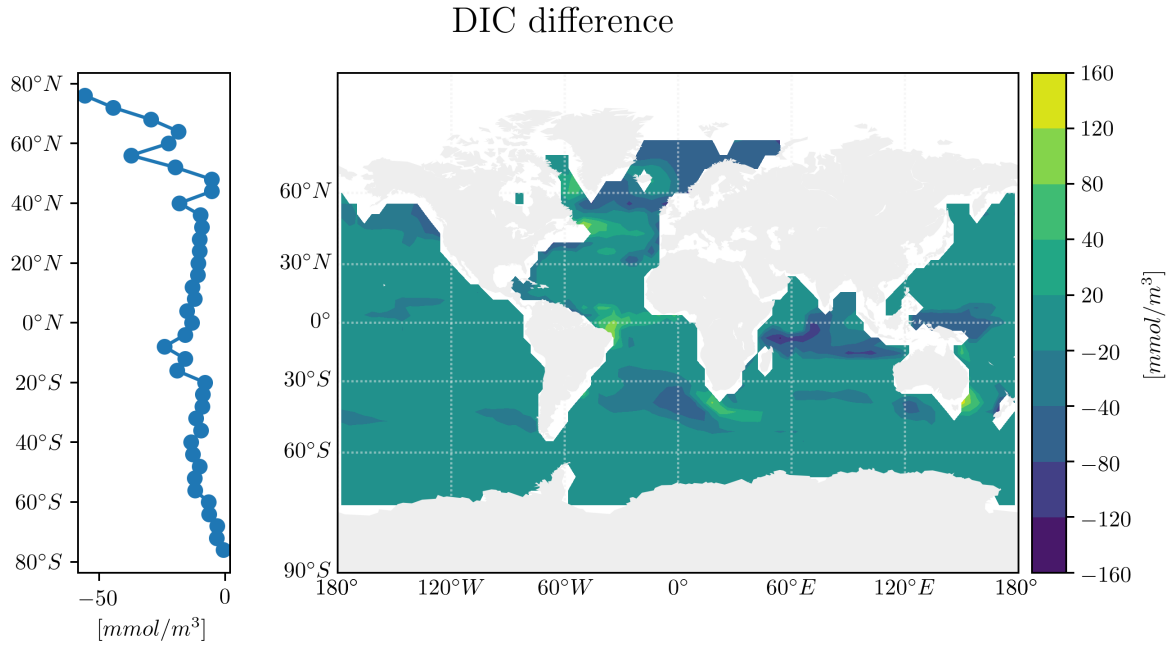


Figure 21: After a collapsed AMOC surface nutrients are reduced in the North Atlantic. Left: Mean difference in DIC concentration by latitude

Reduced surface DIC and temperature both contribute to a reduction in the CO_2 partial

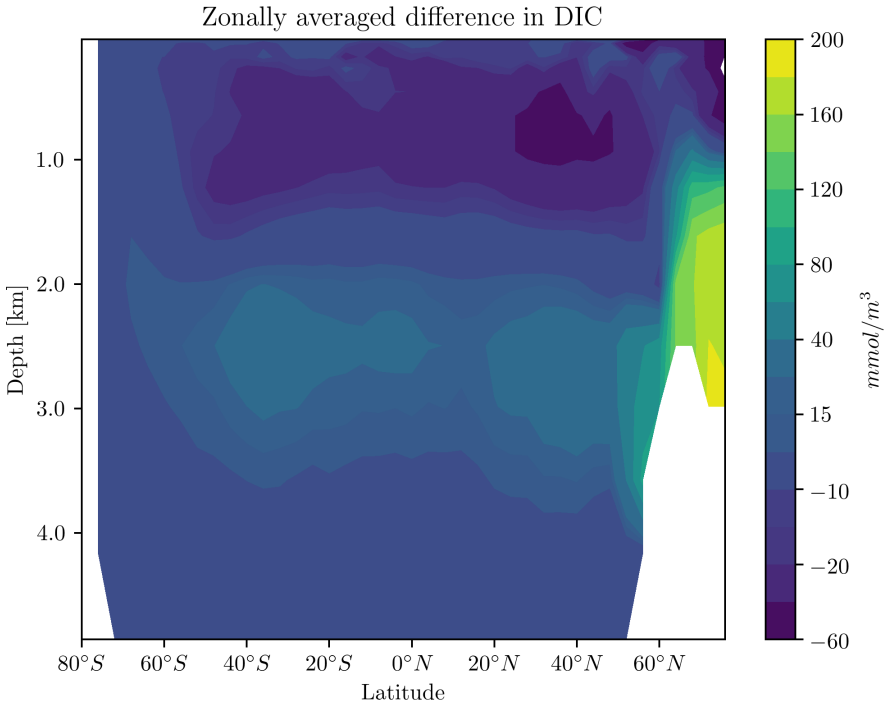


Figure 22: Zonally averaged difference in DIC concentration 200 years after after collapsing the AMOC. The concentration of DIC is reduced in the upper 1km and increased in the abyss.

pressure in the surface layer, which results in an increased flux of CO_2 from the atmosphere into the ocean immediately south of Iceland and outflux southwest of that area at around 45 °N shown in figure 23. This is consistent with results from (Nielsen et al., 2019), although they find an additional outflux North of Scandinavia, which is not present in this simulation, but they did collapse the AMOC by changing the Nothern Hemisphere insolation rather than introducing a freshwater flux. After introducing the freshwater flux and reducing surface salinity, there is an immediate drop in pCO_2 in the affected area. This causes an increased flux of CO_2 into the ocean. Reducing salinity will effect the approximated equilibrium concentrations described in section 5 and therefore the calculated pCO_2 and CO_2-flux. The difference in mean $pCO2$ does reduce with simulation time as a result of larger influx. The chosen parameter set did, as described in section 12, cause an elevated surface phytoplankton population. With the reduction in the plankton population, the additional carbon bound in the plankton is released in the North Atlantic depths causing a larger difference in DIC than other similar experiments.

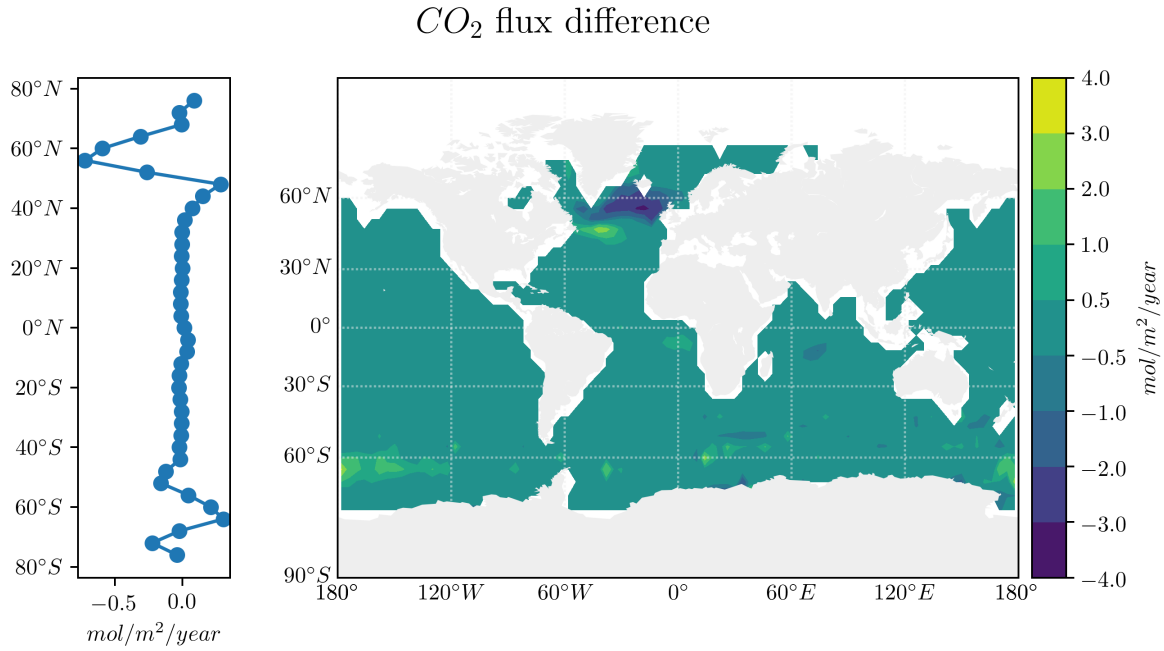


Figure 23: The CO_2 flux after a reduction in AMOC strength transports additional gas into the North Atlantic ocean as a response to the lowered DIC and temperature

14 Conclusion

In this project I have implemented a module for biogeochemical simulations in the general circulation model Veros. The module is designed for ease of use for a general user as well as model developers.
For a general user I introduce a simple configuration file in which the user may select which tracers should be available in the model along with a set of interactions between them. Interactions are represented by rules, which are documented by the developer and specify between which tracers they work, where they work geographically as well as when in the execution order they work. I also introduced sets of rules for common interaction sets, which are enabled the same way as single rules.
Users wishing to extend the functionality of the module are only required to write rules for the dynamics they intend to represent without having to introduce it in every possible configuration tree.

To represent tracers in the model the biogeochemistry module in Veros differs from its main inspiration, the npzd module in UVic ESCM, in that tracers are created as classes which contain attributes relevant to the tracer. Other models maintain separate variables for tracer concentrations and each of the attributes relevant to the tracer. By creating classes for the tracers I allow for creating tracers with similar behavior with different parameters by instantiating objects with the different parameters. Furthermore it is possible to store additional metadata, which would have been difficult to relate to the correct tracer in other models.

The combination of tracers as objects and interactions between them as rules allows Veros to graphically display the active configuration of the model. This is helpful in development as well as tool for displaying intended behaviour. Metadata in objects and rules is presented directly in the graph.

The design allows for building several specialized configurations with rules and tracer classes

as building blocks without introducing configuration checks or duplicate code, which keeps the logical structure readable and maintainable allowing users to focus on implementing features relevant to their needs without risking breaking existing functionality or reimplement the feature for every possible configuration option while keeping the evaluation structure clean.

I used the implemented model to reproduce present day measurements by running a 400 year simulation starting from uniform nutrient distributions and exponentially decreasing plankton distributions. Model parameters were used to fit the output to measurements. That output was used as the base for a simulated collapse of the Atlantic Meridional Overturning Circulation. I conclude a weakening of the AMOC would result in reduced primary production and a redistribution of nutrients from the surface into the abyss.

In conclusion I was able to create a powerful, easy to use biogeochemistry module for Veros, which adheres to the vision of Veros by being accessible, easy to use, easy to verify and easy to modify. The module produced produced results representative of modern day measurements.

A Figures with extended dynamics

Each of these figures represent a configuration option, which includes one or more circuits. They are all working within the logical structure of the model as presented. They are constructed as rules and require no conditionals within the structure.

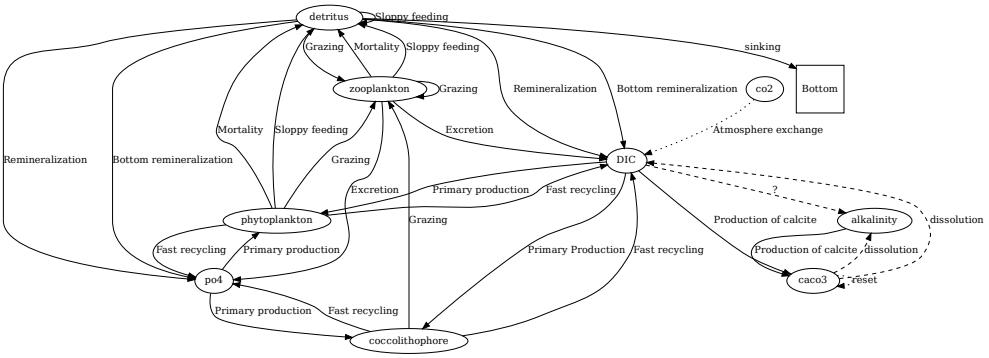


Figure 24: A minimal configuration including coccolithophores, which tracks calcium carbonate explicitly

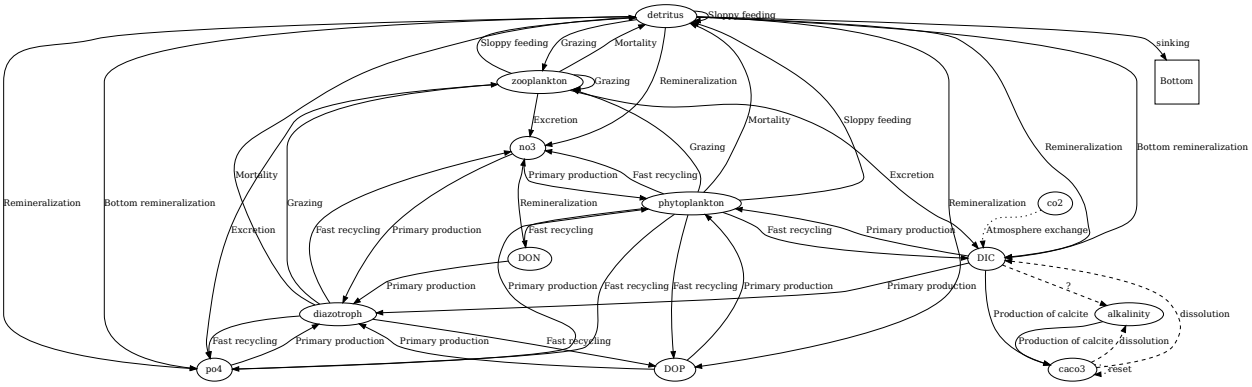


Figure 25: Interaction diagram for a basic NPZD setup including nitrogen fixing diazotrophs. A configuration like this would require selecting different rules for the basic NPZD functionality, but the structure of model evaluation and rule selection remain the same.

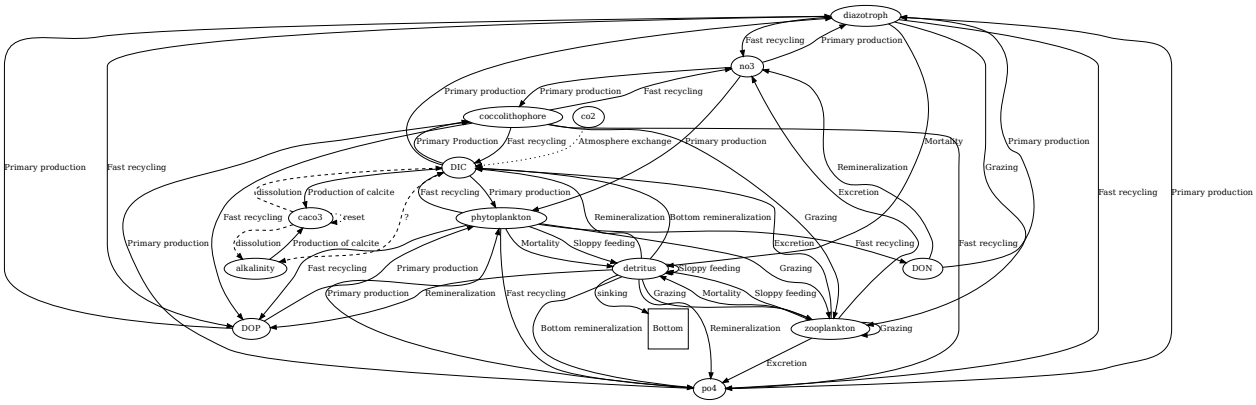


Figure 26: Configured with basic NPZD functionality, a carbon cycle, calcifiers and a Nitrogen cycle.

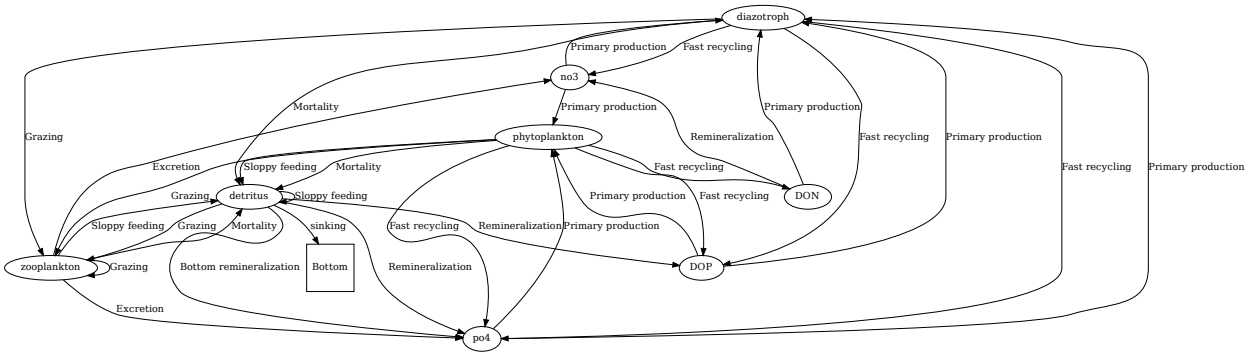


Figure 27: Nitrongen cycle without a carbon cycle. It extends the basic NPZD configuration.

B Simplified overview of evaluation structure

Below is a simplified overview of the biogeochemistry model structure. I have left out the actual calculations in order to provide a readable overview of when and where rules and object methods are evaluated.

```

  # work on a copy of results to avoid overwriting
  tracers = vs.npzd_tracers.copy()

  # Update variables, which only depend on time dependent values from other
  # Veros modules
  calculate_from_model_input(vs.temp, vs.salt)

  # Evaluate pre rules
  pre_results = [rule.function(rule.source, rule.sink) for rule in vs.pre_rules]
  for res in pre_results:
    for key, value in res:
      vs.tracers[key] += value

  # Ensure minimum concentrantions of updated tracers and set flags
  reset_tracers(pre_results.keys)

  # Primary bio loop
  for n in range(n_bio):
    # store copy of swr from veros variable
    shortwave_radiation = vs.shortwave_radiation.copy()

    # Object methods
    for name, object in tracers.items():
      # perform all object methods
      if hasattr(object, 'primary_production'):
        primary_production[name] = object.primary_production()

      if hasattr(object, 'mortality'):
        mortality[name] = object.mortality()

      # etc.

      # block light
      if hasattr(object, 'light_attenuation'):
        shortwave_radiation *= exp(- object.light_attenuation *
        object.cumsum())

      # Calculate import - export for sinking tracers
      if hasattr(object, 'sinking_speed'):
        import_minus_export[name] = calc_export(object)

    # evaluate primary rules
    primary_results = [rule.function(rule.source, rule.sink) for rule in vs.primary_rules]
    for res in primary_results:
      for key, value in res:
        tracers[key] += value * vs.dt_bio

    # add import-export to tracer
    for key, value in import_minus_export:
      tracers[key] += value

    # ensure minimum concentrations
    reset_tracers(primary_results.keys)

  # Evaluate post rules and update tracers
  post_results = [rule.function(rule.source, rule.sink) for rule in vs.post_rules]
  for res in post_results:
    for key, value in res:
      tracers[key] += value

  # reset updated tracers
  reset_tracers(post_results.keys)

  # return only difference in calculations
  return tracers - vs.npzd_tracers


def npzd():
  # Get results from physical transport
  transport_results = transport_tracers(npzd_tracers)
  # Get results from biogeochemistry
  bgc_results = biogeochemistry()

  # Add results to tracers
  for tracer in npzd_tracers:
    tracer += transport_results + bgc_results

  # Ensure minimum concentration
  reset_tracers(npzd_tracers)
```

C Abbreviation list

- NPZD = Nutrients, Phytoplankton, Zooplankton, Detritus
- BGC = Biogeochemistry
- DIC = Dissolved Inorganic Carbon
- DOP = Dissolved Organic Phosphorus
- DON = Dissolved Organic Nitrogen
- PAR = Photosynthetically active radiation
- AMOC = Atlantic Meridional Overturning Circulation
- UVic ESCM = University of Victoria Earth System Climate Model

References

Brown, Nicolas and Eric D. Galbraith (Aug. 18, 2016). "Hosed vs. Unhosed: Interruptions of the Atlantic Meridional Overturning Circulation in a Global Coupled Model, with and without Freshwater Forcing". In: *Climate of the Past* 12.8, pp. 1663–1679. ISSN: 1814-9332. DOI: 10.5194/cp-12-1663-2016. URL: https://www.clim-past.net/12/1663/2016/ (visited on 08/31/2019).

CORE.2 Global Air-Sea Flux Dataset (2008). Boulder CO: Research Data Archive at the National Center for Atmospheric Research, Computational and Information Systems Laboratory. URL: https://doi.org/10.5065/D6WH2N0S.

Dickson, A.G. and C. Goyet (Sept. 1, 1994). *Handbook of Methods for the Analysis of the Various Parameters of the Carbon Dioxide System in Sea Water. Version 2*. ORNL/CDIAC–74, 10107773. DOI: 10.2172/10107773. URL: http://www.osti.gov/servlets/purl/10107773-BfTXAM/webviewable/ (visited on 02/04/2019).

Group, NASA Ocean Biology Processing (2015). *SeaWiFS Level 2 Ocean Color Data Version 2014*. DOI: 10.5067/orbview-2/seawifs_oc.2014.0. URL: http://oceancolor.gsfc.nasa.gov/data/10.5067/ORBVIEW-2/SEAWIFS_OC.2014.0 (visited on 07/24/2019).

Häfner, Dion, René Løwe Jacobsen, Carsten Eden, et al. (Aug. 16, 2018). "Veros v0.1 – a Fast and Versatile Ocean Simulator in Pure Python". In: *Geoscientific Model Development* 11.8, pp. 3299–3312. ISSN: 1991-9603. DOI: 10.5194/gmd-11-3299-2018. URL: https://www.geosci-model-dev.net/11/3299/2018/ (visited on 08/31/2019).

Häfner, Dion, René Løwe Jacobsen, Roman Nuterman, et al. (Sept. 3, 2018). *Veros - Versatile Ocean Simulation in Pure Python*. Version v.0.1.1? Copenhagen University, Niels Bohr Institute. URL: https://github.com/dionhaefner/veros.

Key, R. M. et al. (Dec. 2004). "A Global Ocean Carbon Climatology: Results from Global Data Analysis Project (GLODAP): GLOBAL OCEAN CARBON CLIMATOLOGY". In: *Global Biogeochemical Cycles* 18.4, n/a–n/a. ISSN: 08866236. DOI: 10.1029/2004GB002247. URL: http://doi.wiley.com/10.1029/2004GB002247 (visited on 07/24/2019).

Kuhlbrodt, T. et al. (Apr. 24, 2007). "On the Driving Processes of the Atlantic Meridional Overturning Circulation". In: *Reviews of Geophysics* 45.2, RG2001. ISSN: 8755-1209. DOI: 10.1029/2004RG000166. URL: http://doi.wiley.com/10.1029/2004RG000166 (visited on 05/25/2019).

Kvale, K. F. et al. (May 27, 2015). "Explicit Planktic Calcifiers in the University of Victoria Earth System Climate Model, Version 2.9". In: *Atmosphere-Ocean* 53.3, pp. 332–350. ISSN: 0705-5900, 1480-9214. DOI: 10.1080/07055900.2015.1049112. URL: http://www.tandfonline.com/doi/full/10.1080/07055900.2015.1049112 (visited on 10/13/2018).

Martiny, Adam C, Jasper A Vrugt, and Michael W Lomas (Dec. 2014). "Concentrations and Ratios of Particulate Organic Carbon, Nitrogen, and Phosphorus in the Global Ocean". In: *Scientific Data* 1.1, p. 140048. ISSN: 2052-4463. DOI: 10.1038/sdata.2014.48. URL: http://www.nature.com/articles/sdata201448 (visited on 08/28/2019).

Morris, A.W. and J.P. Riley (Aug. 1966). "The Bromide/Chlorinity and Sulphate/Chlorinity Ratio in Sea Water". In: *Deep Sea Research and Oceanographic Abstracts* 13.4, pp. 699–705. ISSN: 00117471. DOI: 10.1016/0011-7471(66)90601-2. URL: https://linkinghub.elsevier.com/retrieve/pii/0011747166906012 (visited on 07/10/2019).

Nielsen, Søren B. et al. (Apr. 2019). "Two-Timescale Carbon Cycle Response to an AMOC Collapse". In: *Paleoceanography and Paleoclimatology* 34.4, pp. 511–523. ISSN: 2572-4517, 2572-4525. DOI: 10.1029/2018PA003481. URL: https://onlinelibrary.wiley.com/doi/abs/10.1029/2018PA003481 (visited on 05/25/2019).

Oschlies, Andreas and Veronique Garçon (Mar. 1999). "An Eddy-Permitting Coupled Physical-Biological Model of the North Atlantic: 1. Sensitivity to Advection Numerics and Mixed Layer Physics". In: *Global Biogeochemical Cycles* 13.1, pp. 135–160. ISSN: 08866236. DOI: 10.1029/98GB02811. URL: http://doi.wiley.com/10.1029/98GB02811 (visited on 10/13/2018).

Press, William H., ed. (2002). *Numerical Recipes in C: The Art of Scientific Computing*. 2. ed., reprinted with corr. to software version 2.10. OCLC: 249221154. Cambridge: Univ. Press. 994 pp. ISBN: 978-0-521-43108-8.

Schmittner, A., N. Gruber, et al. (Sept. 4, 2013). "Biology and Air-Sea Gas Exchange Controls on the Distribution of Carbon Isotope Ratios $\delta13$C) in the Ocean". In: *Biogeosciences* 10.9, pp. 5793–5816. ISSN: 1726-4189. DOI: 10.5194/bg-10-5793-2013. URL: https://www.biogeosciences.net/10/5793/2013/ (visited on 10/13/2018).

Schmittner, A., A. Oschlies, et al. (Sept. 2005). "A Global Model of the Marine Ecosystem for Long-Term Simulations: Sensitivity to Ocean Mixing, Buoyancy Forcing, Particle Sinking, and Dissolved Organic Matter Cycling: MARINE ECOSYSTEM MODEL". In: *Global Biogeochemical Cycles* 19.3. ISSN: 08866236. DOI: 10.1029/2004GB002283. URL: http://doi.wiley.com/10.1029/2004GB002283 (visited on 10/13/2018).

Schmittner, Andreas (2018). *https://github.com/OSU-CEOAS-Schmittner/UVicMOBI_fct_npzd_o2_n_f* URL: https://github.com/OSU-CEOAS-Schmittner/UVicMOBI_fct_npzd_o2_n_fe_ca_caco3_c13_n15 (visited on 09/03/2018).

Schmittner, Andreas et al. (Mar. 2008). "Future Changes in Climate, Ocean Circulation, Ecosystems, and Biogeochemical Cycling Simulated for a Business-as-Usual CO $_2$ Emission Scenario until Year 4000 AD: FUTURE CHANGES OF CLIMATE, ECOSYSTEMS". In: *Global Biogeochemical Cycles* 22.1, n/a–n/a. ISSN: 08866236. DOI: 10.1029/2007GB002953. URL: http://doi.wiley.com/10.1029/2007GB002953 (visited on 10/13/2018).

Takahashi, Taro et al. (Apr. 2009). "Climatological Mean and Decadal Change in Surface Ocean pCO2, and Net Sea–Air CO2 Flux over the Global Oceans". In: *Deep Sea Research Part II: Topical Studies in Oceanography* 56.8-10, pp. 554–577. ISSN: 09670645. DOI: 10.1016/j.dsr2.2008.12.009. URL: https://linkinghub.elsevier.com/retrieve/pii/S0967064508004311 (visited on 07/25/2019).

Uppström, Leif R. (Feb. 1974). "The Boron/Chlorinity Ratio of Deep-Sea Water from the Pacific Ocean". In: *Deep Sea Research and Oceanographic Abstracts* 21.2, pp. 161–162. ISSN: 00117471. DOI: 10.1016/0011-7471(74)90074-6. URL: https://linkinghub.elsevier.com/retrieve/pii/0011747174900746 (visited on 07/10/2019).

UVic ESCM (2018). Version 2.9. URL: http://www.climate.uvic.ca/model/ (visited on 09/10/2018).

Wanninkhof, Rik (1992). "Relationship between Wind Speed and Gas Exchange over the Ocean". In: *Journal of Geophysical Research* 97.C5, p. 7373. ISSN: 0148-0227. DOI: 10.1029/92JC00188. URL: http://doi.wiley.com/10.1029/92JC00188 (visited on 02/04/2019).

Weiss, R.F. (Nov. 1974). "Carbon Dioxide in Water and Seawater: The Solubility of a Non-Ideal Gas". In: *Marine Chemistry* 2.3, pp. 203–215. ISSN: 03044203. DOI: 10.1016/0304-4203(74)90015-2. URL: http://linkinghub.elsevier.com/retrieve/pii/0304420374900152 (visited on 02/04/2019).