

The Bumper Book of *.muffins*

(The cGENIE.muffin user-manual and
introduction to Earth system modelling)

Andy Ridgwell

Copyright © 2017 Andy Ridgwell

PUBLISHED BY DERPY-MUFFINS INC.

MYMUFFIN.SEAO2.ORG

Licensed under the Creative Commons Attribution-NonCommercial 3.0 Unported License (the “License”). You may not use this file except in compliance with the License. You may obtain a copy of the License at <http://creativecommons.org/licenses/by-nc/3.0>. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

First printing, October 2017



Contents

1	Installation, configuration, basic usage	9
1.1	Before anything else ...	10
1.2	Starting (dozing?) off ...	13
1.2.1	Logging in!	13
1.2.2	Downloading the model code	13
1.2.3	Configuring the code	14
1.2.4	Testing the model code	15
1.2.5	Brief notes on git and code	16
1.3	Running the model	19
1.4	Model output	23
1.4.1	Time-slice output	23
1.4.2	Time-series output	24
1.4.3	File naming convention	24
1.5	Viewing model output	25
1.5.1	Time-series output	25
1.5.2	2- and 3-D time-slice output	25
1.6	Submitting experiment ‘jobs’	27
1.7	‘Restarts’	29
2	Climate dynamics & experimental design	31
2.1	Brrrrrrrrrr – it’s chilly on ... snowball Earth!	33
2.2	Further ideas	37
2.2.1	Feedback loop analysis	37

2.2.2	Continental configuration (1)	37
2.2.3	Continental configuration (2)	38
2.2.4	Geothermal heat input	38
2.2.5	Seasonality	39
3	Ocean circulation	41
3.1	Tracing ocean circulation	43
3.2	Poking the climate beast	47
3.3	Further ideas	50
3.3.1	Hosing investigations	50
3.3.2	'Anti-hosing' investigations	50
3.3.3	Response to transient warming	51
3.3.4	Ventilation age tracers	51
4	Climates of Past Worlds	53
4.1	The climate of the Cretaceous	56
4.2	The climate of the early Eocene	59
5	Alternative Worlds	61
5.1	Generating fake Worlds	62
5.2	Understanding ocean circulation	63
5.2.1	Model investigation over-view	63
5.2.2	Deep-water formation (& baroclinic circulation)	64
5.2.3	Gateways and barotropic flow	65
5.2.4	Analysis	66
6	Fossil fuel CO₂ and 'ocean acidification'	71
6.1	Exploring the consequences of fossil fuel CO ₂ emissions	73
6.1.1	Idealized emissions forcing	74
6.1.2	Historical (real-world!) emissions forcing	77
6.2	Further ideas	80
6.2.1	Assessing the importance of emissions rate	80
6.2.2	Determining thresholds of environmental impact	80
7	Ocean biogeochemical cycles	83
7.1	xxx	85
7.2	Ocean carbon geoengineering	86
7.2.1	Iron fertilization	87
7.2.2	Phosphate fertilization	88
7.2.3	Enhanced weathering	88
7.2.4	Modifying spatial patterns	88

7.3	Further ideas	90
7.3.1	Impacts to look out for	90
7.3.2	Other thoughts and suggestions	90
7.3.3	Further modifications of the biological pump in the ocean	90
8	Marine ecosystems and dynamics	95
8.1	Getting going with ECOGEM	97
8.1.1	Running the model	97
8.1.2	Viewing 2D time-slice output	98
8.1.3	Comparing to observations	98
8.2	Ecosystem configuration	100
8.2.1	Visualising composite data	101
8.2.2	Iron limitation	102
8.3	Increasing ecological complexity	104
8.3.1	Plankton size classes	104
8.3.2	Viewing 2D time-slice output	104
8.3.3	Create your own ecosystem	106
8.4	Build it up, tear it down	107
8.4.1	A fully size-structured ecosystem	107
8.4.2	Ecosystem characteristics	107
8.4.3	Mixotrophy	110
8.5	Ecology in past oceans	112
8.6	Ecology in fake oceans	119
9	The land surface	121
9.1	xxx	123
10	Atmospheric dynamics	125
10.1	xxx	127
10.2	Geoengineering	128
11	Geological carbon processes	129
11.1	The long tail of CO_2 (and other tales from the sediments)	131
11.2	Sediment model output	133
11.3	Quantifying how long is the long tail of CO_2	135
11.4	Sediments of the modern Earth	137
11.5	The marine geology of fake worlds	138
11.6	Further ideas	142

12	Proxies and Reconstructing the Past	145
12.1	xxx	147
13	Synthesis: Earth system dynamics	149
13.1	README	150
13.2	How low can you go?	151
13.2.1	Global weathering rate.	154
13.2.2	Iron fertilization.	154
13.2.3	Remineralization depth.	154
13.2.4	Macro nutrient inventory and uptake.	155
13.2.5	CaCO ₃ :POC rain ratio.	155
13.2.6	Sea-ice extent.	156
13.2.7	Atlantic circulation.	156
13.2.8	Global ocean circulation / ‘brine rejection’.	156
13.2.9	Salinity	158
13.2.10	Terrestrial carbon storage	158
14	muffin model output	161
14.1	Overview (and types of model output)	162
14.2	Time-slice output	165
14.2.1	Frequency and timing of <i>time-slice</i> data saving	165
14.2.2	Experiment end saving	166
14.2.3	Seasonal/monthly data saving	166
14.2.4	More frequent data saving	167
14.3	Time-series output	168
14.3.1	Frequency and timing of <i>time-series</i> data saving	168
14.3.2	Saving orbital insolation	168
14.4	Data field selection	169
14.5	Re-start files	171
14.6	Useful output	172
14.6.1	Physics	172
14.6.2	(Bio)Geochemistry	175
14.6.3	Biology/Ecology	178
15	Introduction to model results analysis	179
15.1	Plotting with Excel	180
15.2	Plotting with Panoply	181
15.2.1	Issues with Panoply default settings	182
15.2.2	Basic plots – examples	183
15.2.3	Difference (anomaly) plots	183
15.2.4	Ocean velocity plots	184

15.3	MATLAB plotting	186
15.3.1	MATLAB 101	186
15.3.2	MATLAB and ASCII (<i>time-series</i>)	186
15.3.3	MATLAB 'Import Data'	186
15.3.4	MATLAB and netCDF (<i>time-slice</i>)	187
16	muffinplot	191
16.1	Installation	192
16.2	Time-series plotting	192
16.3	Spatial plotting	194
16.3.1	Basic usage	198
16.3.2	Example analysis	201
16.3.3	Further refinements	202
16.3.4	Time-series plotting	204
16.3.5	Sediment model output analysis	204
17	muffindata	205
17.1	Data re-gridding	206
17.2	Miscellaneous data processing	207
18	muffingen	209
18.1	muffingen basics	210
18.1.1	overview of the muffingen software	210
18.1.2	Installing muffingen	210
18.1.3	Running muffingen	211
18.2	Configuring muffingen – overview	212
18.2.1	muffingen configuration examples	213
18.3	Configuring muffin model experiments	214
18.4	muffingen parameter settings – details	215
19	FAQ	217
19.1	Installation related questions	218
19.2	Cluster/queue questions	219
19.3	Help! My experiment has died ... why?	220
19.3.1	Other sources of error	220
19.3.2	Meaning of specific error messages	221
19.4	General running and configuring experiments questions	222
19.5	Climate-y questions	225
19.6	Ocean biogeochemistry-y questions	226

19.7	Questions of long-term carbon cycling and stuff	230
19.8	<i>Data saving questions</i>	231
19.9	<i>Forcings questions</i>	232
20	HOW-TO	235
20.1	HOW-TO ... get started with cGENIE.muffin	236
20.2	HOW-TO ... linux	237
20.3	HOW-TO ... git	239
20.4	HOW-TO ... Sun Grid Engine	240
20.5	HOW-TO ... Ubuntu muffin	241
20.6	HOW-TO ... RedHat muffin	245
20.7	HOW-TO ... macOS muffin	246
20.8	HOW-TO ... Windows(!) muffin	249
20.9	HOW-TO ... diagnose how the model works	253
20.10	HOW-TO ... screw with the climate system	258
20.11	HOW-TO ... configure ocean geochemistry	259
20.12	HOW-TO ... biology	261
20.13	HOW-TO ... remineralize ... differently	266
20.14	HOW-TO ... force the system	269
20.15	HOW-TO ... do stuff with sediments	273
20.16	HOW-TO ... geoengineering	285
20.17	HOW-TO ... orbits	290
21	Model Code	297
21.1	Testing	298
21.2	Common Modifications	299
21.3	Uncommon (III-advised) Modifications	301
22	Examples	307
22.1	xxx	308
	Bibliography	309
	Books	309
	Articles	309

Before anything else ...

Starting (dozing?) off ...

Logging in!

Downloading the model code

Configuring the code

Testing the model code

Brief notes on git and code

Running the model

Model output

Time-slice output

Time-series output

File naming convention

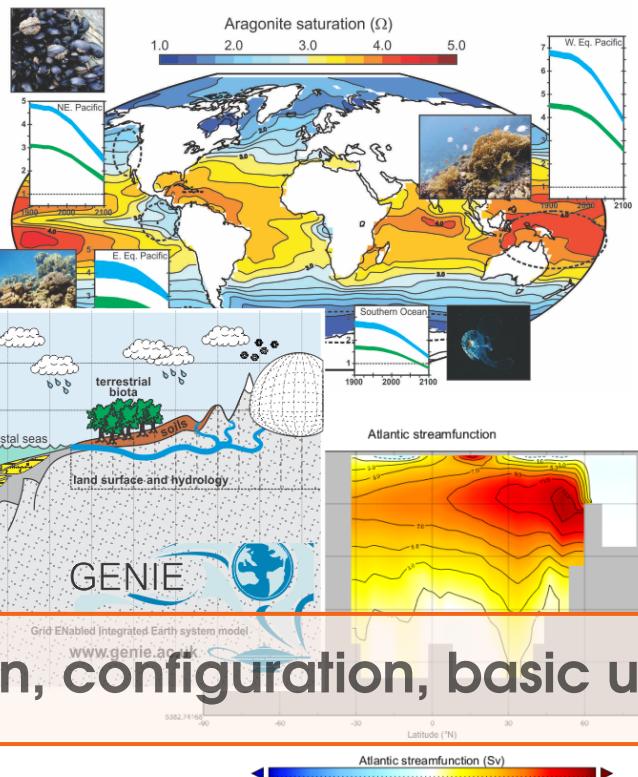
Viewing model output

Time-series output

2- and 3-D time-slice output

Submitting experiment 'jobs'

'Restarts'



1. Installation, configuration, basic usage

Stuff to keep in mind:

- (c)GENIE.)muffin is a model. Models ARE NOT the 'real World'. (Don't get confused!)
- The low resolution (for a 3-D ocean circulation model) of the **muffin** model limits its applicability for very short time-scale problems. In configurations not incorporating the PLASIM atmospheric model, there is no atmospheric dynamics or inter-annual variability in the ocean-atmosphere coupled system.
- **muffin** is best thought of as a 'discovery and exploring' tool for learning how the Earth system (might) work rather than necessarily as a detailed 'simulation' tool.
- It is possible to have fun. (Once the installation pain is over ...)

1.1 Before anything else ...

ReadMe

Some warnings and reminders in this manual are repeated over and over and over and ... over again. Some warnings and reminders are repeated over and over and over and ... over again. This is because you will forget immediately each time! ;)
!¹

Software version naming conventions

You will be using the current version of the cGENIE Earth system model, code-named ‘**muffin**’ (if **Apple** can have ‘Leopard’, ‘Lion’ etc., I can haz a baked goods version naming convention). The documentation may not be fully consistent in this respect ... and you may need to translate occurrences of e.g. a directory named ‘cgenie’ to ‘cgenie.muffin’. For brevity, the **cGENIE.muffin** model will be referenced by just ‘**muffin**’.

linux ...

The (**cGENIE.**)**muffin** model currently compiles and runs only under **linux**² (e.g. distributions such as **Ubuntu**) and unix-like operating systems (such as **macOS**). Furthermore, **muffin** has traditionally been configured and accessed (aka ‘run’) at the ‘command line’ of the **linux** (or **macOS** equivalent, which is unix-like) operating system. The command line is a place where you type text and when you press Return, something (hopefully, good!) happens. Typically the stuff you type started with a ‘command’ word, and often followed by one or more options and parameters. The command word and any options / parameters MUST be separated by SPACES.

The start of each line of the command line is indicated with something like: \$. The \$ is called the ‘prompt’ and is ... prompting you to type some input (commands, Tweets, swear words, etc.). See – the computer is just sat there waiting for you to command it to go do something (stupid?). Typically, you will also be informed (reminded) of the user-name, computer name, and current directory, e.g.:

```
[username@host ~]$
```

which is in this example is user ‘username’ (yours will be different!) on computer name ‘host’³ and the current directory is the ‘home’ directory – represented by the ~ symbol. If, for example, you were instead currently in the myfolder directory, you would see at the command prompt:

```
[username@host myfolder]$
```

If you are not or not very familiar with the **linux/unix** command line, such as how to navigate up and down the directory tree and to display the contents of the current directory you are in – for a brief summary of some basic/useful **linux** commands and usage – see Section 20.2 (page 237).

NOTE: BE VERY CAREFUL that spaces are not missed out when typing out example lines. Also be careful not to confuse the number one (1) for the letter el (l). Mis-spelling/typing will probably be the primary reason for any wailing and gnashing of teeth ...

¹Also read footnotes please.

²For some of you, the mechanics of running the model will be about as much fun as sticking your tongue in an electrical outlet (a popular hobby in England). (However, if you are an experienced linux/unix/tongue-in-electrical-socket user, you can skip onto the next Section and save yourself an entire 15 seconds of reading words.)

³Sprout the cat will eventually appear under ‘cat-of-the-day’ on my home-page if you press F5 enough times – all my computing clusters are named after my cats ...

NOTE-the-second: In places, instructions may be given for specific programs and computer platforms and hence may differ slightly from the software reality in front of you. Use your judgement in translating such instructions. Many other alternative software choices exist for editing files or viewing results, as are other ways of configuring software and file editing/transferring methodologies. Do what suits you best – you can view such instructions where they occur, as more representing an example methodology rather than a literal interpretation of the Constitution.

Required computer hardware

It is possible to install and run the ‘(cGENIE.)**muffin**’ Earth system model either on a linux box (e.g. **Ubuntu**) or on a **Mac**⁴⁵. Otherwise, you will need an account on a linux-based server or cluster. In this case, you can use any platform (**linux**, **macOS**, **WindoZ**, as well as **iOS** and **Android** (which is based on **linux** in any case)) to connect to the cluster (but see software requirements, below).

Required computer software

To install and run **muffin**, at a minimum are needed⁶:

1. A FORTRAN (f90 and f77 combatable) compiler of some sort.

This may come with the operating system as standard, possibly **gfortran**.

2. A **git** client.

If not standard, this is relatively easy to add and install.

3. Compiled **netCDF** libraries (not so much fun ...).

To edit files and visualize results, you will need some specific software. The exact software will depend on your operating system, but essential are:

1. A viewer for netCDF format spatial data. A **Java** viewer called **Panoply** is provided by NCAR for all platforms – <http://www.giss.nasa.gov/tools/panoply/> (Note that you will need **Java** installed!)

(Or alternatively: **MATLAB**.)

2. A simple text editor, except not the rubbish default **Windows** one – you need one that can display **unix** ASCII text without screwing it up. Options for **Windows** users are: **notepad++** (<https://notepad-plus-plus.org/>) **SciTE** (<https://www.scintilla.org/SciTE.html>) (**linux** and **Mac** users need no special/different editor compared with your standard editor – everything will display just fine). You can also use **linux** command line based editors such as **vi**.

If operating **muffin** remotely (e.g. on a computing cluster), the you also will need:

1. A terminal (‘shell’) window. This is no problem for **linux** and **Mac** users (you already have one built in). For **Windows**, either download a simple (and old) **SSH** client (ssh-client) from my website⁷ or you can get hold of e.g. **PuTTY** (<http://www.putty.org/>).

2. A sftp (secure file transfer) client for convenience (i.e. dragging and dropping files between local and remote computers, and opening files directly on the remote computer cluster). If you have installed ssh-client (**Windows**, above) then a sftp client is already included as part of this software. If using **PuTTY** (**Windows**) you might try downloading **WinSCP** (<http://winscp.net/eng/index.php>). For **macOS**, you can connect to the server through

⁴Sets of detailed installation instructions are available in the HOW-TO section of this manual.

⁵Note that it is not possible at this time to run **muffin** under **Windows** (at least, not without near infinite pain).

⁶Only if **muffin** is being run locally and a cluster account is not provided (but assuming that your cluster account is correctly configured ...).

⁷<http://www.seao2.info//cgenie/software/ssh-client.exe>

the **Terminal**, but some sftp software for viewing/navigating server file structure include: **FileZilla** (recommended), **Cyberduck**, **TextWrangler**. For linux, maybe **FileZilla**.

File editors ...

You will need to edit text-based configuration files, possibly in installing and configuring **muffin**, but definitely in configuring model experiments. So now might be a good time to check that you can use the/an editor! (You will also be using the same editor to view some of the model output.)

You have two alternative options for editing and viewing text files, depending on whether you are a **unix** nerd with no life, or prefer anything to do with computers to be wrapped in cotton wool and covered with dollops of treacle. EITHER: Use the linux **vi** (**vim**) application (or similar e.g. **emacs**) if you are familiar with it. I think that this pretty much sucks as a text editor and life is far too short and brutal if you don't like this sort of thing ... OR ... use a suitable linux-friendly text editor (NOT **Micro\$oft Notepad**) in conjunction with the **Secure File Transfer Client**. For example: **SciTE** (<http://www.scintilla.org/SciTE.html>) is suitable, or **Notepad++**.

If you fiddle about with the settings under Options/Preferences in the **WinSCP** program and apply a little common sense, it should be possible to configure things so that you can simply double-click on a file in the remote (right-hand) window panel and it will open like magic (almost)! Saving the file after editing) should then result in the file being saved back to the cluster. Or you can select Edit With (and then **SciTE**) from right-mouse-button-clicking on the filename. Or ... a crude but workable approach is to use an sftp client to drag the file to your local machine (assuming **muffin** is installed remotely), edit it there, and then drag it back again.⁸

Model documentation in general

This, and additional documentation (of varying degrees of up-to-date-ness) can be found:

1. On [GitHub](#).

Here, the **latex** source for the documentation lives, allowing you to compile the most up-to-date PDF document. And ... make changes yourself and have them incorporated into the official documentation⁹.

2. On my [website](#).

Here, are only compiled, PDF versions of the documentation are provided, and may be somewhat out-of-date.

This document in particular

The instructions in general may not be bug-free – use your judgment.

Go!

OK – now we are ready to start ...

⁸Note that care still has to be taken to avoid certain **Microsoft** text editing programs under **Windoze**.)

⁹First clone the **git** repository. Make changes. Commit them locally. Make a 'pull request' ...

1.2 Starting (dozing?) off ...

You are going to be installing the model from scratch – why? Why not? It will be a happy character-building experience for you ... trust me ...

1.2.1 Logging in!

Log into your **linux/macOS** box!¹⁰

Or ... if you are running **muffin** remotely (e.g. on a cluster account), log into the remote account from a suitable terminal¹¹:

- If logging in via a **linux/macOS** box, open a terminal/shell window and simply SSH in¹², e.g.

```
$ ssh username@clusternname
```

where `clusternname`, the cluster (or remote server) name(!) might, for example, be `catname.ggy.bris.ac.uk`, and enter your account password (and tell it whether or not you want this password stored, if asked).

- On a **Windoz** machine – first start e.g. the **WinSCP** program (an sftp file transfer client) or **PuTTY** (a ‘shell’, or ‘terminal’ window). Under ‘Host Name’, enter the `clustername` (e.g. `catname.ucr.edu`):

The ‘Port number’ should be set to 22. Enter your computing cluster user-name on the line below this (‘User Name’) and then the Password. Click on Login. This is your file transfer client.

You will also need a terminal window (assuming you did not open that first). This can be opened by clicking on the ‘Open session in PuTTY’ icon on the top icon row, or pressing `Ctrl+P`.

You should now have TWO windows open – a ‘shell’ window (lines of text on an otherwise blank screen) and a file manager (transfer) window. Ensure that you have both these before moving on. It is recommended that you maximize both these windows to full screen. (But no-one will die horribly for not doing so. Probably ...).

You can also log in directly from the shell/terminal window of e.g. **PuTTY** rather than first opening an sftp connection.

1.2.2 Downloading the model code

The next step is to download a copy of the source code for **muffin**.

As of 2018, the branch of the (c)GENIE source code that is **cGENIE.muffin**, was migrated from a server hosted by the University of Bristol, to GitHub (and hence the versioning system changed from **svn** to **git** – a “distributed” version control systems (DVCS). The address of the new (centralized) home for **muffin** is:

<https://github.com/derpycode/cgenie.muffin>

¹⁰If you fail at this step, you’ll have to take up box-modelling instead.

¹¹It very much depends on what software you are using. Provided are instructions for some examples, but only examples, and your reality may be rather different.

¹²If your current directory looks something like this: `[username@clusternname ~]$` then you are probably already logged in! Otherwise, it will look like: `username@localcomputername:~$`

IMPORTANT: if you had previously installed a version of **muffin** under **svn**, note that the default name of the installation (`cgenie.muffin`) is the same.¹³

From here, there are 2 ways to get your mitts on the model code:

1. By downloading an archive file, containing all the code etc. For this – click on the green Clone or Download button, and select Download ZIP.
You then unpack/unzip the files and directory structure where you want it.
This [archive download] is a perfectly workable way to proceed ... as long as you neither want to update the code with whatever new developments or bug fixes occur in the future, nor want to have any code changes you might make, become part of the official **muffin** code (i.e. it becomes a one-off installation that has no connection to the GitHub repository).
2. The preferred/advised way is to *clone* the repository to where you intend to run **muffin**¹⁴. While you can also use a GUI based git client, easiest is at the command line, using the command `git clone`¹⁵:

```
$ git clone https://github.com/derpycode/cgenie.muffin.git
```

By doing this, you have created your own code repository (and an identical copy of the one hosted on GitHub). As part of the `git clone` command, you also automatically *check out* (from your very own personal repository) a copy of the code. Note that the major difference then with the **svn** system, is that previously, the GENIE code repository existed only on the University of Bristol server, and you *checked out* the code remotely from there.

1.2.3 Configuring the code

You now need to configure some local environment settings so that **muffin** can find all the libraries etc. that it needs. The changes you need to make will depend on the platform where you will be running **muffin** (i.e. where you have just cloned the code repository to).

- **Ubuntu.** This is the default assumed platform. No changes are necessary assuming that the netCDF libraries are installed in their default locations and a relatively recent version of netCDF is used.¹⁶ And you have also jumped the shark with a successful install of other libraries and softwares ... see Section 20.5.
- **domino** (UCR cluster). There is a single file to edit (which lives in the `genie-main` directory):
 1. In `user.mak` – at the end of the file, the netCDF path needs to be changed. First comment out¹⁷ the default setting (with a # symbol) and then un-comment¹⁸ the line

¹³If you already have a previous (**svn** version) installation of **muffin**, i.e. in a directory `cgenie.muffin`, you should rename, or simply delete (`rm -f -r cgenie.muffin`) the original installation so as not to lose files or risk ending up with a mixed-up corrupted installation. For example:

```
$ mv cgenie.muffin cgenie.muffin.svn
```

will rename the installation directory, adding the extension `.svn` on the end. You will no longer be able to run the original installation now, but you can fix this by editing some of the directory *paths* – see FAQ.

¹⁴But see later for other/better ways of working.

¹⁵This: "... clones a repository into a newly created directory, creates remote-tracking branches for each branch in the cloned repository, and creates and checks out an initial branch that is forked from the cloned repository's currently active branch."

¹⁶By 'relatively recent' – the default settings assume that the FORTRAN and C netCDF libraries are separate.

¹⁷To 'comment out' a line – simply add a # symbol to the very start of the line. When **muffin** runs, this line will be ignored.

¹⁸To 'un-comment' a line – simply remove (delete) the # symbol from the beginning of the line.

under the heading `### domino ###`.

- **eevee** (UCR cluster). There are 2 files to edit (all of which live in the `genie-main` directory):
 1. In `user.mak` – as per for `domino` (except the `user.mak` line needed is headed `### eevee ###`).
 2. In `makefile` – as a consequence of a currently incomplete netCDF install, missing are the C libraries for netCDF. This affects only a C program that is compiled to complete the tests (by comparing model output with an archived copy of the expected results). For now, this test must be disabled by:
On line 18 – comment out (with a #); the `nccompare` part of that line.
- **sprout** (UoB cluster). There are 2 files to edit:
 1. In `user.mak` – at the end of the file, the netCDF path needs to be changed. First comment out the default setting (with a # symbol) and then un-comment the line under the heading `### sprout ###`.
 2. In `makefile.arc` – towards the end of the file, under the heading:
`# === NetCDF paths ===`
 uncomment the line under:
`### FOR COMBINED C+FORTRAN NETCDF LIBRARIES #####`
 and comment the 2 lines under:
`### FOR SEPERATE C AND FORTRAN NETCDF LIBRARIES ###` (so as to select, combined, rather than sperate, netCDF libraries).
- **almond** (UoB cluster). As per for `sprout` (except the `user.mak` line needed is headed `### almond ###`).
- **macOS**. Please refer to the separate macOS instructions in Section 20.7.

1.2.4 Testing the model code

Finally, you need to test the code to ensure that all the files have been cloned/installed correctly. First, change directory (see: Figure 1.1) to¹⁹:

```
cgenie.muffin/genie-main
```

If you are not ‘linux-friendly’ (see: Section 20.2 for **linux basics**) – maybe at first do this in steps – list the contents of the directory (`ls`) to check where you are (i.e. what directories are available to chance to), then change to `cgenie.muffin` (`cd cgenie.muffin`), then list again (`ls`) (and see what further directories are there), then change to `genie-main` (`cd genie-main`), and only then ... type:

```
$ make testbiogem
```

This compiles a basic carbon cycle enabled configuration of **muffin** and runs a short test, comparing the results against those of a pre-run experiment (also downloaded alongside the model source code). It serves to check that you have the software environment correctly configured. There may be some ‘Warnings’ reported (== somepony’s sloppy programming) but these are not detrimental to the ultimate science results (we hope!).

‘Success’ of this test is indicated by:

```
**TEST OK**
```

¹⁹Note: the model is *always* run from `cgenie.muffin/genie-main`

You can then be certain that the model you have installed is producing identical (within tolerance) results to everyone else in the World who has ever installed **muffin**. Note that the model will pause for a long time at the line:

```
./genie.job -t -k -f configs/eb_go_gs_ac_bg_test.xml -o /home/genie00/cgenie_output
-c /home/genie00/cgenie -g ../../cgenie -m "" > testbiogem.out;
```

This is quite ‘normal’ – the model is thinking! Also – ignore the compiler warnings . . . (reflecting my lack of adequate software engineering skills).

If the test doesn’t ‘work’²⁰ – try issuing the command:

```
$ make cleanall
```

and then re-try the test. Refer to the FAQ section at the end of this book for further clues as to non-working model installations.

That is it as far basic installation goes. But . . .

GitHub does not host files larger than 100 MB, and the ‘lookup table’ for calculating opal dissolution in sediments (see e.g. Ridgwell et al. [2003]) is larger than this. It has hence been committed to the git repo as an archived file, and if a (open) silica cycle is to be employed in **muffin** experiments, it needs to be unpacked. A script is provided for this. From `genie-main`:

```
$ ./installmuffin.sh
```

will unpack the opal lookup table to `genie-sedgem/data/input` as well as unpacking a copy of the calcium carbonate lookup table²¹.

1.2.5 Brief notes on git and code

The basic git clone

The simplest workable installation of **muffin**, as described above, is to use `git clone`. You end up with a carbon copy of the muffin code repo (I am too lazy to type out ‘repository’ again) which you have also automatically now ‘checked out’. Changes and developments will occur to the code on the GitHub repo from time-to-time, and it may be that either you might benefit by using a more up-to-date code base, or specific changes may have been made that you absolutely need. To determine the status of your repo, type²² (from `cgenie.muffin`):

```
$ git status -uno
```

However, git has not actually compared your repo with the one on GitHub, because this is apparently network expensive (as if you don’t spend the rest of your life killing the internet by streaming Rick and Morty). So, use:

```
$ git fetch
```

which will ‘download [new and modified] remote content but not update your local repo’s working state’. If you now type `git status -uno`, git can tell you if there is newer content (e.g. ‘Your branch is behind ‘origin/master’²³). To merge in the fetched content:

²⁰Note that if you have disabled the compilation of the C program that compares netCDF files, the test will run but never complete and you’ll not get a **TEST OK** reported, even if the test experiment ran correctly.

²¹For now, the CaCO_3 lookup table also included in the git repo in its expended (unpacked) form.

²²Here, the `-uno` ensures that files (e.g. experiment configuration files) you have created, but not added and committed, are not listed.

²³`origin` is the origin of the repo – GitHub, and `master` is the name of the branch, in this case, the default branch name.

```
$ git merge
```

Both these commands are also combined in a single command²⁴:

```
$ git pull
```

If you are not developing code, and hence not editing files in the repo (but e.g. only adding new model configuration files), your life should mostly be trouble-free with regards to updating your code via pull.²⁵

²⁴If there are no changes (to *fetch* and *merge*), the you get the message: `Already up-to-date.`

²⁵ One exception is, if any of the model installation/configuration files that you might have edited, i.e. one or all of: `genie-main/user.mak`, `genie-main/makefile.arc`, and/or `genie-main/makefile`, have also changed on `origin/master`, then `pull` (or `merge`, after `fetch`) will fail (with the message: `'Please, commit your changes or stash them before you can merge. Aborting'`). The root issue is a conflict between remote file changes, and local ones that you had not committed to your repo.

You probably want to keep your local configuration changes, otherwise you'll probably end up re-doing them all over again. One solution is to sneakily 'hide' (*stash*) them out of sight, by: `$ git stash` You can now pull, updating your repo with respect to `origin/master`. Then – you want your changes back, so apply the stashed changes: `$ git stash apply` If unlucky, there will be conflict as your stashed changes are merged back onto your local repo branch (master). If so, you'll need to edit the file, deciding which line(s) is correct, and delete the version you do not want, along with the ASCII tags/labels.



Figure 1.1: Directory structure of the **muffin** model. Highlighted in red are directories and sub-directories that you will need to access at some point. Vertical green lines designate directory levels, with example commands shown for moving between them.

1.3 Running the model

The overall sequence of configuring and running **muffin**, is shown in Figure 1.2.

At the command-line (\$) in the genie-main directory (not your home directory), you will be entering in a command (`./runmuffin.sh`) together with a list of parameters that will be passed to the model, and as if by magic the model will run (or sometimes not). The form of the command you are going to be issuing is:

```
$ ./runmuffin.sh #1 #2 #3 #4 (#5)
```

(don't type it yet!)

The form of the command requires that you must list at least 4 parameters after `./runmuffin.sh`, separated by S P A C E S and on a single continuous line (even if it ‘wraps’ around across 2 lines of the screen). These parameters are:

#1 ... is the name of the required base (or ‘basic’) configuration (‘*base-config*’) of the model.

#2 ... is the name of the subdirectory (if any) containing the user configuration (‘*user-config*’) file (i.e., the file containing the specification of a particular experiment).

#3 ... is the name of the experiment itself. There must exist a file in the directory specified by parameter #2 (LABS) with exactly the same name as you enter here for parameter #3 (i.e. parameter #3 points to a file in the directory given by parameter #2).

#4 ... is the run length of the experiment in years – this must be entered as an integer.

There is also one optional (5th) parameter (described later).

As an example of running the **muffin** Earth system model:

#1 : The base config is: `cgenie.eb_go_gs_ac_bg.worbe2.BASE`

#2 : The user config directory is: LABS

#3 : The user config file (the experiment name) is: LAB_0.EXAMPLE.

#4 : Run the experiment for ten years: 10

#5 : (There is no restart file, and so no 5th parameter needs to be passed ...)

The full command for your first example experiment, which you are going to issue from the `~/cgenie.muffin/genie-main` directory, then looks like:

```
$ ./runmuffin.sh cgenie.eb_go_gs_ac_bg.worbe2.BASE LABS LAB_0.EXAMPLE 10
```

(you can try it now!)

REMEMBER: This must be entered on a single CONTINUOUS LINE. The (single) S P A C E S are vital. Take care not to confuse an el (‘l’) with a one (‘1’) when typing this in ... (it is a ‘one’ here).

What should happen is: First, you will end up twiddling your thumbs a while, as all the components of **muffin** are compiled from the raw source code (**FORTRAN**). When it has finished doing this, the model will initialize and carry out some brief self-checking. Only then will it start actually ‘running’ and doing something, starting with a header describing the columns of numbers that follow:

model year – ... guess!
 ice(%) – global sea-ice fraction (%)
 <SST> – global sea surface temperature ('SST') °C
 <SSS> – global sea surface salinity 'SSS' (‰)

The choice of what information to display on screen as the model is running is rather arbitrary, but the chosen metrics do tend to summarize some of the main properties of the climate system and carbon cycle – for my own personal convenience rather than reflecting any fundamental scientific truth ... you may also see columns of information for:

pCO₂(umatm) -- mean atmospheric CO₂ concentration (in units of μatm)
 $\delta^{13}\text{CO}_2$ – mean $\delta^{13}\text{C}$ value of atmospheric CO₂ (‰) (NOTE: only if ^{13}C tracer is selected)
 <DIC> – global mean ocean dissolved inorganic carbon (DIC) concentration ($\mu\text{mol kg}^{-1}$)
 <ALK> – global mean ocean alkalinity (ALK) concentration ($\mu\text{eq kg}^{-1}$) and in experiments with a modern continental configuration, also:

- AM0(Sv) – Atlantic meridional overturning circulation (Sv)

This information is reported at the same intervals as time-series data (see later and/or refer to the User Manual) is saved and is indicated by:

```
>>> SAVING BIOGEM TIME-SERIES AVERAGE CENTERED @ year :
```

Interleaved between these lines are lines reporting the saving of time-slice data (the 2- and 3-D model states – more of which later as well as in the User Manual). These appear as:

```
>>> SAVING BIOGEM TIME-SLICE AVERAGE CENTERED @ year :
```

You can stop the model at any point (all data up to that time will have been saved) by hitting: <Ctrl-C> (CONTROL key + 'C' key).

Just from examining the screen output: how close to steady state does the system appear to have come after just 10 years? i.e., do SST and/or sea-ice extents appear to be converging towards stable (constant) values? This will be an important question to think about later on: 'has the model reached steady-state (and does it matter)?'

In this example, the output should look something like the following:

```
*****
*** Initialisation complete: simulation starting ...
*****
```

model year	*	pCO2(uatm)	d13C02	*	AMO(Sv)	ice(%)	<SST>	<SSS>	*	<DIC>(uM)	<ALK>(uM)
\$N\$	0.00	278.000	-6.500		0.000	0.000	-0.000	34.900		2244.000	2363.000
>>>	SAVING BIOGEM TIME-SLICE AVERAGE CENTERED @ year :								0.500		
>>>	SAVING BIOGEM TIME-SERIES AVERAGE CENTERED @ year :								0.500		
\$N\$	1.00	279.960	-6.598		13.613	0.744	2.509	34.901		2241.498	2363.111
>>>	SAVING BIOGEM TIME-SLICE AVERAGE CENTERED @ year :								1.500		
>>>	SAVING BIOGEM TIME-SERIES AVERAGE CENTERED @ year :								1.500		
\$N\$	2.00	279.525	-6.580		12.828	3.499	4.471	34.901		2240.173	2363.135
>>>	SAVING BIOGEM TIME-SERIES AVERAGE CENTERED @ year :								2.500		
\$N\$	3.00	279.258	-6.568		11.695	5.028	5.996	34.901		2239.169	2363.161
>>>	SAVING BIOGEM TIME-SERIES AVERAGE CENTERED @ year :								3.500		
\$N\$	4.00	279.044	-6.558		10.444	5.929	7.209	34.901		2238.354	2363.191
>>>	SAVING BIOGEM TIME-SLICE AVERAGE CENTERED @ year :								4.500		
>>>	SAVING BIOGEM TIME-SERIES AVERAGE CENTERED @ year :								4.500		
\$N\$	5.00	278.899	-6.551		9.380	6.191	8.156	34.902		2237.664	2363.220
>>>	SAVING BIOGEM TIME-SERIES AVERAGE CENTERED @ year :								5.500		
\$N\$	6.00	278.777	-6.545		8.500	6.623	8.975	34.902		2237.069	2363.246
>>>	SAVING BIOGEM TIME-SERIES AVERAGE CENTERED @ year :								6.500		
\$N\$	7.00	278.680	-6.541		7.922	6.629	9.637	34.903		2236.548	2363.267
>>>	SAVING BIOGEM TIME-SERIES AVERAGE CENTERED @ year :								7.500		
\$N\$	8.00	278.601	-6.537		7.917	6.738	10.225	34.903		2236.087	2363.285
>>>	SAVING BIOGEM TIME-SERIES AVERAGE CENTERED @ year :								8.500		
\$N\$	9.00	278.528	-6.534		7.952	6.740	10.732	34.904		2235.682	2363.301
>>>	SAVING BIOGEM TIME-SLICE AVERAGE CENTERED @ year :								9.500		
>>>	SAVING BIOGEM TIME-SERIES AVERAGE CENTERED @ year :								9.500		
\$N\$	10.00	278.466	-6.531		8.025	6.694	11.176	34.904		2235.325	2363.314

```
*****
*** Simulation complete: shutdown starting ...
*****
```



Figure 1.2: Schematic of the sequence-of-events in configuring and running an experiment.

1.4 Model output

The first thing to note about output (i.e., saved results files) from **muffin** is that every science module saves its own results in its own sub-directory (and sometimes in very different and difficult-to-fathom ways ...) – see Figure 1.1. All the sub-directories of results, plus copies of input parameters and the model executable, are gathered together in a directory that is assigned the same name as the experiment (== *user-config* file name). The experiment results directories all live in:

```
~/cgenie_output
```

and will be assigned a directory name something like:

```
LAB_0.EXAMPLE
```

(this being the results directory name for an experiment called LAB_0.EXAMPLE). Within this directory are each module's results sub-directories. We will primarily consider only results saved by the ocean biogeochemical module '**BIOGEM**' (subdirectory: `biogem`). The results files in this example will thus be found in:

```
~/cgenie_output/LAB_0.EXAMPLE/biogem
```

BIOGEM has a flexible and powerful facility of saving results by means of spatially explicit ‘time-slices’, and as a semi-continuous ‘time-series’ of a single global (or otherwise representative mean) variable. In contrast, the atmospheric chemistry module '**ATCHEM**' does not save its own results (**BIOGEM** can save information about atmospheric composition and air-sea gas exchange) while the marine sediment module **SEDGEM** does save its own results, but only at the very end of a model experiment (**BIOGEM** can also save the spatial distribution of sediment composition as time-slices as well as mean composition as a time-series). Furthermore, to attain a common format for both ocean physical properties and biogeochemistry, **BIOGEM** can save a range of ocean results in addition to temperature and salinity, such as: velocities, sea-ice extent, mixed layer depth, convective frequency, etc.

1.4.1 Time-slice output

One of the most informative data sets that can be saved is that of the spatial distribution of properties (such as tracers or physical ocean attributes). However, saving full spatial distributions (e.g., a $36 \times 36 \times 8$ array) for any or all of the tracers each and every time-step is clearly not practical; not only in terms of data storage but also because of the detrimental effect that repeated file access has on model run-time. Instead, **BIOGEM** will save the full spatial distribution of tracer properties only at one or more predefined time points (in units of years). These are termed *time-slices*. At the specified time points, a set of spatially-explicit data fields are saved for all the key tracer, flux, and physical characteristics of the system. However, rather than taking an instantaneous snapshot, the time-slice is constructed as an average over a specified integration interval (the default is set to 1.0 years, i.e. an annual average). **BIOGEM** assumes that the specified time point represents the mid-point of the (annual) average with the results that output years end up being reported as e.g.,

0.5

1.5

2.5

4.5

...

(the mid-points of averages made over the intervals: 0-1, 1-2, 2-3, 4-5 years, etc.).

1.4.2 Time-series output

The second data format for model output is much more closely spaced in time. Model characteristics must then be reducible to a single meaningful variable for this to be practical (i.e., saving the time-varying nature of 3-D ocean tracer distributions is not). Suitable reduced indicators would be the total inventories in the ocean and/or atmosphere of various tracers (or equivalently, the mean global concentrations / partial pressures, respectively). Like the time-slices, the data values saved in the time-series files represent averages over a specified integration interval (the default is set to 1.0 years (annual average) but the results are reported with respect to the mid-point of the average which is where the ‘.5’ bits come in again).

1.4.3 File naming convention

The biogem results directory will contain files with names of the form:

- `_restart.nc` (is the re-start file created from the run you have just complete, and can be ignored).
- `biogem_series_*.res` – these are the time-series files (in ASCII / plain text format).
- `biogem_year_*_diag_GLOBAL.res` – these contain (global diagnostics) summary information and are saved at the same frequency as the time-slices (also as ASCII / plain text).
- `fields_biogem_2d.nc` – 2-D fields of ocean and atmosphere properties, as NetCDF.
- `fields_biogem_3d.nc` – 3-D fields of ocean properties, as NetCDF.

1.5 Viewing model output

1.5.1 Time-series output

A descriptive summary of all the time-series (`biogem_series_*.res`) data files is given in the **muffin** User Manual if you are really that bored. The files of most immediate use/relevance are:

- `biogem_series_atm_humidity.res` - mean atmospheric (surface) humidity
- `biogem_series_atm_temp.res` - mean atmospheric (surface) air temperature
- `biogem_series_misc_opsi.res` - min/max overturning stream-function values (e.g. AMOC)
- `biogem_series_misc_seaice.res` - mean ocean sea-ice cover and thickness
- `biogem_series_ocn_sal.res` - mean ocean surface and whole ocean salinity
- `biogem_series_ocn_temp.res` - mean ocean surface and whole ocean temperature

One way of viewing the contents of files is to change directory to the experiment results directory and opening the file in the file editor. But that is not so much fun.

Instead – change to the experiment results directory and then to the `biogem` sub-directory in the Secure File Transfer Client, and try double-clicking (if you have set up the **WinSCP** preferences correctly) or right-mouse-button-clicking (the then Edit with) on one of the `.res` files (listed above). For `biogem_series_ocn_temp.res`, you should see 2 columns – time and mean (whole) ocean temperature (°C). (However, in subsequent exercises a fuller output will be created with additional columns, with one for mean surface ocean temperature ‘SST’ (°C) as well as mean benthic (bottom water) temperature (°C)). Other results files may differ in the numbers of columns but all should be identifiable from the header information.

Note: **WinSCP** does not automatically refresh the directory listing. If you cannot see the results sub-directory with the experiment name you have just run, 99 times out of 100, it is because the display of the **WinSCP** needs to be refreshed – there is an icon at the top of the program window or hit the ‘F5’ key.

For your information and edification (only): **Excel**, or **MUTLAB** if you prefer, can be used to graph the time-series results. Either way you will have to deal with the header line(s) that are present at the top of the file (and preceding the rows of data).

In **Excel**: Choose File then Open. You will want to select Files of Type ‘All Files (*.*)’. In the Text Import Wizard window you can request that **Excel** skips the first few lines to start the import on the 2nd or 3rd line of the text file. Alternatively: set an appropriate column width manually in **Excel** to ensure that the columns of data are correctly imported.

MUTLAB will ignore lines starting with a %, which the time-series starts with. However, it may be that the header line wraps-around and there is in effect a 2nd header line but without a %. In this case, extra care (or a quick edit of the header in the ASCII file) will be required to load the data into **MUTLAB**.

1.5.2 2- and 3-D time-slice output

For the time-slice NetCDF (*.nc) files you will be using a program called **Panoply**. If you want your own (FREE!) copy of this utility, you can get it here (and is available for: **Windoz**, **Mac**, and linux operating systems): <http://www.giss.nasa.gov/tools/panoply/>.

When you open the NetCDF file, you will be presented with a ‘Datasets and Variables’ window (on the left hand side of the application window). This contains a list of all the parameters available that you can display. You will find that the ‘Long Name’ description of the variable will be the most helpful to identify the one you want. Simply double-click on a variable to display. For the 3-D fields you will be asked first whether you want a ‘Longitude-Latitude’ or ‘Latitude-Vertical’ plot (for the 2-D fields, the plot display will immediately open). For the ‘Longitude-Latitude’ plots – there are multiple levels (depth layers) in the ocean - these data that can be plotted from the surface to the abyssal ocean. For the ‘Latitude-Vertical’ plots – there are multiple possible longitudes at which to plot slices. The default is the global mean meridional distribution. There is also an option for ‘Longitude-Vertical’ plots (which we will not use). For all three: there may be multiple time-slices (i.e., you can plot data saved from different years). You can interpolate the data or not (often you may find that it is clearer not to interpolate the data but to leave it as ‘blocky’ colors corresponding to the resolution of the model), change the scale and colors, overlay continental outline, change the projection, etc etc. Grey cells represent ‘dry’ grid points, i.e., continental or oceanic crust.

NOTE: The default settings in **Panoply** can mislead:

1. By displaying the very 1st time-slice (often year mid-point 0.5) time-slice rather than the experiment end. (This can confuse as it can look like an experiment has not done anything!)
2. By interpolating the data (not always misleading). To remove interpolation, un-tick: ‘Interpolate’ in the ‘Arrays’ tab.
3. By displaying a global zonal mean by default when selecting Latitude-Vertical plots. Then, to further confuse you, by plotting the output up-side-down (to invert: in the ‘Grid’ tab, hit ‘Swap B/T’ (for swap bottom/top).
4. By listing all ‘Plottable variables’ (option at the bottom of the window), when what you *ideally* want are the shorter and less confusing list of ‘Georeferenced variables’.
5. In Longitude-Latitude plots, by overlaying the modern continental output. (**muffin** land is marked in grey.)
6. By fitting a scale to the plot when the display window is opened, but not changing the scale when e.g., time or depth is changed. (The point of confusion is that you can quickly move outside the scale and end up with all model points dark blue or red.) Re-fit the scale, or manually set limited, in the ‘Scale’ tab. So be careful when opening a new plot that you are looking at what you *think* you are looking at ... All the defaults can be changed via the ‘Edit’ drop-down menu and ‘Preferences’.

Explore different data fields and play with different ways of displaying them. Aim for a set of display properties that show the information you are interested in / want to present in the clearest possible manner. Try different years (time-slice number), depth level (for a Latitude-Longitude plot), or longitude (for a vertical section).

To save plots in **Panoply**:

File

Save Image As ...

Then select the location, filename, and graphics format.

1.6 Submitting experiment ‘jobs’

This bit is no particular fun at all, but it is a very handy ‘trick’ for running the model in the background, and maximizes drinking time in the bar vs. sat bored watching a computer screen :)

Running jobs interactively is all very well, but there are three important limitations: (1) The connection between your terminal and the server computer running the model must remain unbroken. Anything more than a fleeting loss of internet connectively may result in the experiment terminating. (2) You can only run one experiment at a time ... unless you want to have thousands of separate terminal open ...? I thought not ... (3) Any cluster or computer you are likely to be accessing using a shell will not have many computing cores itself, either because it is a single machine with only one or two processors, or if a cluster, by using a terminal you are running on the ‘head node’, which will have similar computing core limitations to running on a single machine. The more experiments you run simultaneously, the slower they will all run ...

The alternative is to submit your experiment as a ‘job’ to a queuing system which then manages what compute resources are used to run the model. Once you have submitted the experiment, that is it – you can go straight to the pub :)

For example – to run the same experiment as before (LAB_0.EXAMPLE) for maybe 100 years (or even longer if you wish – I am just pulling factors of 10 out of thin air here) but now submit the experiment as a job to the cluster queue, type:

```
$ qsub -q dog.q -j y -o cgenie_log -V -S /bin/bash runmuffin.sh
cgenie.eb_go_gs_ac_bg.worbe2.BASE LABS LAB_0.EXAMPLE 100
```

(Again: SINGLE, CONTINUOUS LINE.) Here, the particular queue name is **dog**.

Note that now you should omit the ‘./’ bit before **runmuffin.sh**. (If you are interested (I know that you are not): the options following **qsub** and before **runmuffin.sh** do things like re-directing screen output and error messaging to a file and specify which linux ‘shell’ to assume. It is even possible to receive an email when the job is done :)) The status of the cluster queue and how your experiment job is getting on (e.g., “Is it finished yet?”) can be checked by typing:

```
$ qstat -f
```

(**qstat -f -u “*”** will show all jobs on the cluster.)

After submitting an experiment, you receive a job number. This number appears in the first column in the queue status information when you issue a **qstat -f** command. You should see your job appear on one of 6 compute nodes, numbered 0-0 to 0-5), although it might briefly reside as a ‘PENDING JOB’. For each node, there are multiple processing cores (depending on the specific cluster and queue), meaning that multiple instances of **muffin** can run simultaneously on each node. For an 8-level ocean based configuration of **muffin**, being run for 100 years, the job should remain there in the queue for a few minutes before ‘disappearing’ (your clue that it has finished, or died²⁶ ...). If you periodically re-issue a **qstat -f** command you can follow your job’s progress.

A rough rule of thumb is that 8-level ocean **muffin** @ a horizontal grid resolution of 36x36 will simulate about 1000 years per CPU hour. The 16-level version (which you will use later), runs at about 300-400 years per CPU hour.

²⁶If your experiment appears on the queue but vanishes after a few seconds, it has most likely died :(

NOTE: It may be that the **FORTRAN** compiler is not accessible by the computer nodes. The implication of this is that *the muffin executable must be already compiled BEFORE a job is submitted to the queue.*

In other words; if you have just changed the model resolution or continental configuration, or number of tracers (i.e. changed the *base-config*) or issued a `make cleanall` command you MUST briefly run your desired experiment (or equivalent) interactively (i.e., in the shell window) to ensure that everything is correctly compiled. For instance, either run the experiment for a couple of years or start the experiment for the desired full duration, but 'kill it' (Ctrl-C) once the experiment is running successfully.

1.7 'Restarts'

Not much fun here either ... but again – an important and time-saving (== increased drinking time!) modelling technique to learn to use.

By default, model experiments start from ‘cold’, i.e., the ocean is at rest and uniform in temperature and salinity while the atmosphere is uniform in temperature and humidity. All biogeochemical tracers in the ocean have uniform concentrations and/or are zero and there are no biogenic materials in deep-sea sediments. From this state it will take several thousand years (kyr) for the climate system to reach steady-state, and closer to 5 kyr (or more) for ocean biogeochemical cycles and atmosphere CO_2 to reach steady-state, and exceeding 100 kyr for sediment composition to re-balance weathering ... Reaching this the equilibrium state is called the ‘*spin-up*’ phase of the model. There is evidently little point in repeating the *spin-up* for each and every model experiment that are similar except in a single detail (e.g., testing a variety of different CO_2 emissions scenarios all starting from current year 2012 conditions). A facility is thus provided for requesting that a ‘*re-start*’ is used – starting a new experiment from the end of a previous one, usually a *spin-up* that has been run explicitly for the purpose of generating a starting point (*re-start*) of the system at steady-state (equilibrium) for subsequent experiments to continue on from. It is important to note that there is nothing special about a *re-start* – it is simply an experiment that you have already run. Equally, there is nothing special about the *re-starts* you will download next – these you could have generated yourself – it simply saves time to have them provided.

To experiment with using a *re-start*, you will first need to download a file that has been created (a pre-run 10,000 year spin-up). To fetch this: Change to the `cgenie_output` directory (perhaps by going ‘home’ first (`cd <Enter>`), and then changing to `cgenie_output` – refer to linux commands HOW-TO and Figure 1.1), and type:

```
$ wget http://www.seao2.info/cgenie_output/LAB_0.SPIN.tar.gz
```

This downloads an archived/compressed copy of the restart from a location on the interweb. Extract the contents of this archive by typing:

```
$ tar xfzv LAB_0.SPIN.tar.gz
```

Finally, change directory back to `cgenie.muffin` and then `genie-main` so that you are ready to run the model (the model is *always* run from `cgenie.muffin/genie-main`).

A *re-start* can be requested in running an experiment by setting the 5th and last (optional) parameter when entering in the `runmuffin.sh` command. A spin-up of the climate state is provided: `LAB_0.SPIN` which you have just unpacked to the `cgenie_output` results output directory. Create a new (*user-config*) experiment configuration file in the: `/cgenie.muffin/genie-userconfigs/LABS` directory by using the given file `LAB_0.EXAMPLE` as a template (no parameter changes need to be made yet). You can make copies (`cp` command) of the experiment configuration files (e.g., `LAB_0.EXAMPLE`) and assign them different names, e.g., `twilight`, `rarity`, `applejack` (ideally, use a name that helps you remember what the experiment did). Or ... if you and the linux command lines are simply not BFFs, you can drag the *user-config* file (`LAB_0.EXAMPLE`) you want to use as a template to your local PC (/Mac) directory in the file transfer window, rename it either in the file transfer program or e.g., via the Windoz file-manager

(e.g., to LAB_0.NEW), and drag it back to the cluster directory ²⁷ again. You specify the use of the restart state by adding the restart experiment name as the 5th parameter, e.g.:

```
$ ./runmuffin.sh cgenie.eb_go_gs_ac_bg.worbe2.BASE LABS LAB_0.NEW 100 LAB_0.SPIN
```

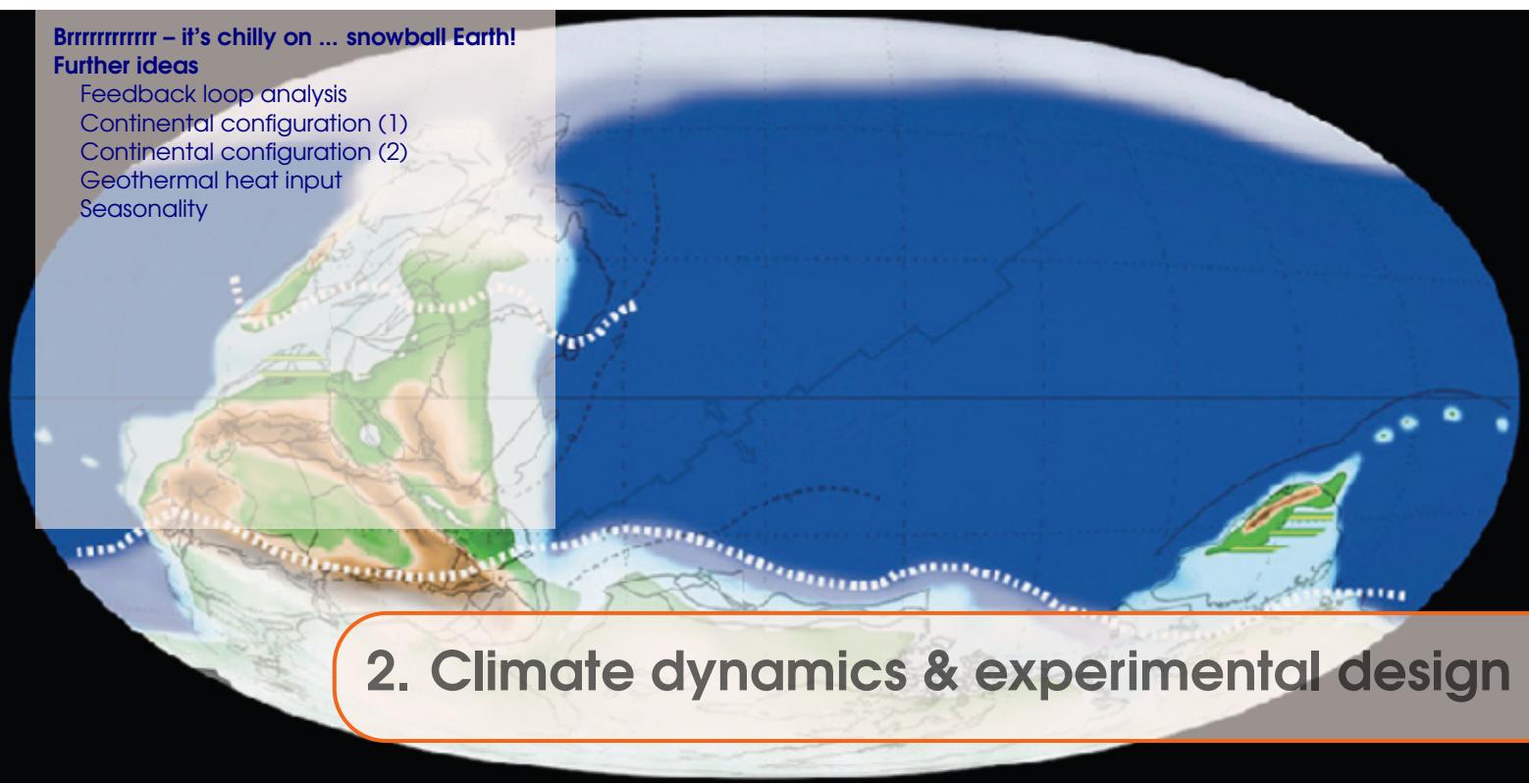
The run-time output should now look noticeably different. There should be no (or perhaps just very little) drift in any of the various variable values outputted to the screen – this is because you have (re-)started from the end of a run that had already ready an equilibrium, steady-state.

²⁷ /cgenie.muffin/genie-userconfigs/LABS

Brrrrrrrrrrr – it's chilly on ... snowball Earth!

Further ideas

- Feedback loop analysis
- Continental configuration (1)
- Continental configuration (2)
- Geothermal heat input
- Seasonality



2. Climate dynamics & experimental design

Stuff to keep in mind:

- Models ARE NOT the ‘real World’ (it is going to be pretty obvious this is the case here).
- Don’t believe what you read in Nature or Science.

Readme

You will need to download a new *restart* file prior to embarking on the snowball Earth experiments. To fetch this: change to the cgenie_output directory, and type (or copy and paste carefully from this PDF ...):

```
$ wget http://www.seao2.info/cgenie_output/LAB_1.SPIN.tar.gz
```

This downloads an archived/compressed copy of the experiment LAB_1.SPIN – effectively, just an experiment (spin-up) that has been run for 5,000 years for you. Extract the contents of this archive by typing:

```
$ tar xfzv LAB_1.SPIN.tar.gz
```

A new experiment results directly will then appear as if you had just run the entire 5,000 year experiment yourself, and you could in fact have done so (remember to refresh the **WinSCP** directory view window if you are using this particular software, or it might appear that nothing has been extracted).

You'll then need to change directory back to **genie-main** to run the model.

2.1 Brrrrrrrrrr – it's chilly on ... snowball Earth!

To illustrate how ‘easy’ it can be to configure an Earth system / climate model such as **muffin** and explore the behavior of the Earth system and its response to perturbation – you are going to induce an extreme cooling of the climate system and see what happens. Solar output was weaker during the late Neoproterozoic, a time when the Earth experienced a series (2 ish) of extreme glaciations. Thus, having a mild climate state to start with must have been dependent on sufficient CO_2 and/or CH_4 in the atmosphere and hence presumably highly elevated compared to the modern World ... sort of opposite to the problem we have today ...

You are going to be running experiments in a similar manner to before, and using the *re-start* experiment that you downloaded:

```
$ ./runmuffin.sh cgenie.eb_go_gs_ac_bg.woreq1.NONE LABS LAB_1.EXAMPLE 100 LAB_1.SPIN
```

But ... rather than use the provided experiment configuration file LAB_1.EXAMPLE (which is provided for you), why not get into the habit of creating new and uniquely named *user-config* files (no harder than copying it and renaming it!). If you keep using the same experiment name, the results will be over-written each time, while having 2 (or more) experiments running simultaneously with exactly the same name causes havoc as they try and over-write each others results files in a somewhat entertaining but ultimately useless way.

Overall: your task in this exercise will be to determine the radiative forcing (or pCO_2 equivalent) threshold required to drive the climate system into a full ice-covered ocean (snowball Earth) state. (Read the *Hyde et al.* [2000] paper.)

Useful 2-D (netCDF—Panoply) variables to view are surface air temperature and sea-ice extent (and/or thickness). Ocean surface temperature and salinity can be viewed in the 3-D NetCDF results file.

Time-series (ASCII .res files) are useful for providing simple mean indicators of global climate such as global ocean fractional sea-ice covered.

Note that the model configuration of an idealized super-continent, positioned symmetrically about the Equator, is pretty unrealistic. But the further you go back in time, the more uncertain it becomes as to exactly where and in what orientation the continents were. Sometimes modelers have to resort to somewhat idealized experiments if the uncertainties are too great. In addition, one can conduct sensitivity experiments to test whether the continental configuration is important to the results. For instance, *Hoffman and Schrag* [2002] discuss the potential importance of continental configuration, while the entire hypothesis of *Donnadieu et al.* [2004] rests on specific details of the continental configuration being realistic.

For this configuration, the solar constant is set weaker than modern to reflect the fact that the Sun’s output has increased with time and during the Neoproterozoic the solar constant would have been ca. 5% weaker. This is set by the model *parameter*:

```
ma_genie_solar_constant= 1285.92
```

which is set at the top of the provided *user-config* file. (For reference, the modern value is $1368 W m^{-2}$.)

Other questions to think about with regards to numerical modeling (and this experiment) are:

- (Is the model configuration and experimental design ‘realistic’ ... ?)
- What is ‘missing’ in the model and what might the implications for your predictions and conclusions be? For example, there is no land-surface scheme (and hence no concept of ‘snow’) in this particular configuration.

- Are the simulations being run for sufficiently long? Why not if not (i.e., justify your choices of parameter values and experimental assumptions)? How might the results and conclusions be biased (if at all)?
 - How would you test model predictions and your overall conclusions?
 - How could the experimental design be improved?
-

To search for the atmospheric CO_2 concentration (or rather, radiative forcing equivalent) that would lead to a ‘snowball Earth’ state in the Neoproterozoic and answer the question: ‘How low does CO_2 have to be to trigger a ‘snowball’?’ you are going to edit the file that controls the specific details of the experiment – the *user-config* file. From the *genie-userconfigs/LABS* directory, open one of the snowball experiments in your preferred text editor. At the top of the file you should see something like:

```
#  
#  
# --- CLIMATE -----  
#  
...  
# scaling for atmospheric CO2 radiative forcing, relative to 278 ppm  
ea_radfor_scl_co2=20.0
```

Each line that is not commented out (i.e., no #) contains a *parameter* name and assigned value pair, with the format:

PARAMETER=VALUE

The value of each parameter can be edited to form a new experiment. (Additional parameter value specifications can also be added, or existing ones deleted.) In this example, the line:

ea_radfor_scl_co2=20.0

specifies a radiative forcing of climate by CO_2 equivalent to x20 modern ($20 \times 278 = 2560$ ppm). If you instead wrote:

ea_radfor_scl_co2=1.0

this would give you a modern (x1, or $1 \times 278 = 278$ ppm) radiative forcing. (Technically: the pre-industrial CO_2 value rather than ‘modern’ *per se*.) Note: CO_2 is not being explicitly modeled in this experiment, but the long-wave radiative forcing associated with a specified concentration of CO_2 (in ratio to modern) is being set instead.

Edit the value of **ea_radfor_scl_co2** (lower or higher) and save the file. Re-run the experiment to see whether sea-ice extent is approaching a new steady state. You may want to try even longer simulations (than 100 years) if it becomes clear that the model is still far from steady-state. You can judge how close to equilibrium things have got by following (and/or plotting) the evolution of e.g., global surface air temperature or sea-ice extent (both time-series files).

HINT: Submitting the experiments to the cluster will allow you to run many many experiments (i.e. each with a different radiative forcing value) simultaneously.

For each experiment you want to be assessing how far towards the Equator the sea-ice limit encroaches through some of the *time-series* and *time-slice* files or even the on-screen summary lines (assuming running interactively rather than via a job submission to the cluster queue). Informative

time-series variables include (but not necessarily be limited to): atmospheric temperature and sea-ice cover. (Sea-ice thickness, on account of the simple physics in the model, low resolution and long time-step, can fluctuate a little in area and volume at times.)

For the *time-slice* data: atmospheric and ocean surface temperature and sea-ice extent (2-D biogem **NetCDF** file) may be informative.

HINT: Be careful with the ‘Fit to data’ scaling feature in **Panoply** – at near complete sea-ice cover, you may find Panoply scaling min and max sea-ice between 99.1 and 99.9% or something. Specific fixed scale limits (e.g. 0 and 100) can be set instead.

In answering the question (‘How low does CO_2 have to be to trigger a ‘snowball’?’), think about what an appropriate degree of accuracy might be for your experiments. Just because computer models generally calculate to around 16 significant places of precision, does not mean you have 16 significant figures of realism. For instance – how many significant figures is the solar constant quoted to and what do you think is the uncertainty in this? Harder to judge is how the assumed (incorrect) continental configuration creates additional uncertainty, or the simple physics assumed in the ocean or sea-ice, or lack of snow on land ...

Once you are happy about the controls on the snowball threshold try and answer the supplementary question:

How high does the (CO_2) radiative forcing have to be in order to escape from a snowball?

Having determined the appropriate radiative forcing value required to create a snowball state, you can use that experiment as a *re-start*, and hence carry out a series of experiments with increasing radiative forcing, all starting from the same snowball climate state you have just created. Defining the radiative forcing / climate path going out of a snowball would complete the hysteresis loop of *Hyde et al.* [2000]. Note that a good *re-start* is one for which the experiment did not sit too long in the snowball state before finishing (the more sea-ice thickness you create in the first experiment, the more you are going to have to melt in the next ...). To achieve this, you can fine-tune the number of years the experiment is run, i.e. having determined the appropriate radiative forcing value required to create a snowball state, find out when in the experiment the snowball state first occurred, and then run a new experiment that finishes only a decade or so after the snowball is initiated.¹

HINT: If you are having trouble deciding whether or not the snowball is heading in the right direction (i.e. towards an exit!), e.g. because sea-ice is always reported at 100% (or close to), you can keep track of whether there is net melting or net freezing by following mean sea-ice thickness (m) as reported in the `biogem_series_misc_seaice.res` time-series output file. Your indication of a melting snowball state is a progressive decline in mean thickness (a proxy for global ice volume). Note that you can open and review the results of *time-series* files at *any* time during the experiment as the lines are written while the data for each time point is generated.

Overall: think critically about the model configuration, the experimental design, and the nature of the scientific question (based on your background reading of snowball Earth). Some of the exploration/testing suggestions (above) may not necessarily give substantially different results. Such a finding would be as valid and interesting as determining an important dependence of a certain assumption, and would for instance indicate that the associated paleo uncertainties are not critical to model assessment of the question.

¹You cannot select when the *re-start* is saved – it is always saved at the end of an experiment.

Always be prepared to justify all your choices for experimental design and model settings, e.g., range of radiative forcing assessed, continental configuration(s), solar forcing, use of re-starts (if any), run duration, etc. etc. etc.

2.2 Further ideas

2.2.1 Feedback loop analysis

To quantify the snowball Earth hysteresis loop in **muffin** as per Figure 2 in *Hyde et al.* [2000] you will need to extract from the model ‘meaningful’ measures of climate (e.g., global surface air temperature, fractional sea-ice coverage) as a function of CO_2 multiples, CO_2 concentration, or (better) radiative forcing. For the latter, in **muffin**, the radiative forcing for a doubling of CO_2 is set at: $5.77 Wm^{-2}$. See: *Myhre et al.* [1998] (Geophys. Res. Lett. 25, 2715–2718) and/or *IPCC* [2007] for more on what radiative forcing is and how it is related to a relative change in CO_2 concentration. Also, for making a comparison with *Hyde et al.* [2000] – for going into the snowball, note that they plot the change in radiative with a ‘cooling’ as positive (a bit daft). Their baseline radiative forcing state (an anomaly of $0Wm^{-2}$) you might assume is equivalent to 278 ppm and hence ~ 130 ppm is an approximately halving of CO_2 and hence creates $\sim 5Wm^{-2}$ of cooling. (You might prefer to plot the radiative forcing change as warming being positive, which makes rather more sense ...)

For coming out of a snowball, because the CO_2 and hence radiative forcing threshold is so high as compared to going in, you may want to be creative in the plotting (assuming attempting to combine both thresholds into a single plot) and, for instance, one might break the scale between the low radiative forcing interval spanning going in and the high one spanning coming out.

Another example is as per Figures 3 and 4 in *Stone and Yao* [2004] (Clim. Dyn. 22, 815–822) (although here it is the solar constant rather than long-wave radiation forcing that is being varied). So in fact, you could try varying the solar constant as an alternative to radiative forcing and hence be able to come up with a plot directly comparable to *Stone and Yao* [2004].

2.2.2 Continental configuration (1)

It was mentioned earlier that the position of the continents is an area of modelling uncertainty and might be important. You can test for this. Four alternative *base-configs* are provided, each defining a different continental configuration:

1. `cgenie.eb_go_gs_ac_bg.wopol1.NONE`
– a single polar super-continent, with an ocean resolution of 36x36 with 8 vertical levels.
(Note potential ‘l’ and 1’1 confusion in ‘wopol1’.)
2. `cgenie.eb_go_gs_ac_bg.wopol2.NONE`
– one continent at each pole, with an ocean resolution of 36x36 with 8 vertical levels.
3. `cgenie.eb_go_gs_ac_bg.woreq1.NONE`
– a single Equatorially-centred super-continent, with an ocean resolution of 36x36 with 8 vertical levels. [current configuration]
4. `cgenie.eb_go_gs_ac_bg.woreq2.NONE`
– two continents straddling the Equator, with an ocean resolution of 36x36 with 8 vertical levels.

You can use the given *user-config* file (`LAB_1.EXAMPLE`) as an experiment template, and any of the alternative configurations can be run very similarly to as per before, i.e.:

```
$ ./runmuffin.sh cgenie.eb_go_gs_ac_bg.xxxxxx.NONE LABS LAB_1.EXAMPLE 100
```

Note that you are using a different *base-config* file name: `cgenie.eb_go_gs_ac_bg.xxxxxx.NONE` where `xxxxx` is one of: `wopol1`, `wopol2`, `woreq1`, or `woreq2`.

Also note that no re-starts are provided for any of these configurations. You may (or may not) want to create some (you will need to judge for yourselves how long to run the restart experiments

for to achieve as close to steady-state as you think is ‘sufficient’). Recall again, that *re-starts* are just ‘normal’ experiments that have already been run. Be careful that when changing from one *base-config* to another, the model re-compiles. Simply running the new configuration briefly is sufficient to ensure this. Experiments can then be safely submitted to a cluster queue, i.e. do not try and submit an experiment using a different *base-config* straight to the cluster queue without having run it (or a short version of the experiment you want) interactively first (to ensure the model is re-compiled). This is also good practice – checking that a new sort of experiment and/or model configuration works as you intend and without hiccups.

2.2.3 Continental configuration (2)

Although much useful can be learned from conceptual configurations and Worlds regarding climate dynamics, it is invariably aesthetically more ‘pleasing’ to also test ideas in a more paleogeographically realistic configuration. Provided is a set of *base-config* and *user-config* files (plus associated boundary conditions) for the position of the continents and climate 635 millions years ago (635 Ma). For this:

The *base-config* is named: `muffin.C.fm0635cb.NONE`

The *user-config* is: `muffin.C.fm0635cb.NONE.SPIN`

NOTE that the *user-config* file is now found in the directory:

`~\cgenie.muffin\user-configs\PALEO`

so that you either need to specify this different (PALEO) directory when running **muffin**, or copy the *user-config* file into LABS.

A *re-start* experiment is provided called `muffin.CB.fm0635cb.NONE.SPIN` and which can be downloaded as per before:

```
$ wget http://www.seao2.info/cgenie_output/muffin.CB.fm0635cb.NONE.SPIN.tar.gz
```

and unpacked by:

```
$ tar xfzv muffin.CB.fm0635cb.NONE.SPIN.tar.gz
```

(Remember: you should be in the `cgenie_output` directory when you do this downloading and unpacking.)

To run (e.g. for 100 years), following on from its *re-start* (and leaving the *user-config* in its PALEO directory):

```
$ ./runmuffin.sh muffin.C.fm0635cb.NONE PALEO muffin.C.fm0635cb.NONE.SPIN  
100 muffin.CB.fm0635cb.NONE.SPIN
```

(all on one line)

NOTE that the *base-config* and *user-config* filenames start `muffin.C.` whereas the *re-start* filename starts `muffin.CB.` ... just to try and trip you up ...

2.2.4 Geothermal heat input

Finally, **muffin** will fairly happily build up sea-ice, apparently without limit (with the remaining wet ocean becoming progressively colder and more saline). In the real world, one might expect some sort of limit to the maximum thickness achieved as the heat diffusion across a progressively greater thickness of sea-ice approaches the heat input at the bottom of the ocean from geothermal energy. Different modes of ocean circulation are also possible if one considers heating from the

bottom as well as cooling (and brine rejection) from the top and which might affect the entry into or exit from a snowball state.

In the experimental setup you have been given, a geothermal heat input is specified in the ocean circulation module via the following :

```
bg_ctrl_force_GOLDSTEInTS=.TRUE.  
bg_par_Fgeothermal=100.0E-3
```

The parameter `bg_par_Fgeothermal` sets the geothermal flux in units of W m^{-2} . (Note that in the Neoproterozoic, the geothermal heat flux could have been somewhat higher than modern. How much higher? A question for **Google** ... ?)

An appropriate research question might be to determine in radiative forcing *vs.* geothermal space (and requiring a 2D grid of parameter combinations to be created and submitted to the cluster), the equilibrium sea-ice thickness and region in which a snowball solution is not possible. However, more simply and suitable to a short exercise: How much of a difference, to the estimated entry and exit thresholds of radiative forcing, does the inclusion of a geothermal input make? E.g., what happens if you set it to zero? What about 10 times modern (or more, although *extreme* seafloor heating can cause numerical instability and the model to crash)?

2.2.5 Seasonality

By default, the idealized model configurations are non-seasonally forced (by solar insolation). You can switch to a seasonally-forced to model by adding the following lines to the *user-config* file:

```
ea_dosc=.true.  
go_dosc=.true.  
gs_dosc=.true.
```

The scientific question here in trying this would be whether or not taking into account a seasonally-varying climate substantially affects the entry (and/or exit) thresholds for a snowball climate state. (At least, whether it is important in the context of the resolution and physics of the model you are using.)

You can also save the data seasonally if you like – see Section 12.2.3 in the **muffin** User Manual (this document!).²

²For reference, your configuration has 24 time-steps per year set for the **BIOGEM** module.

Tracing ocean circulation

Poking the climate beast

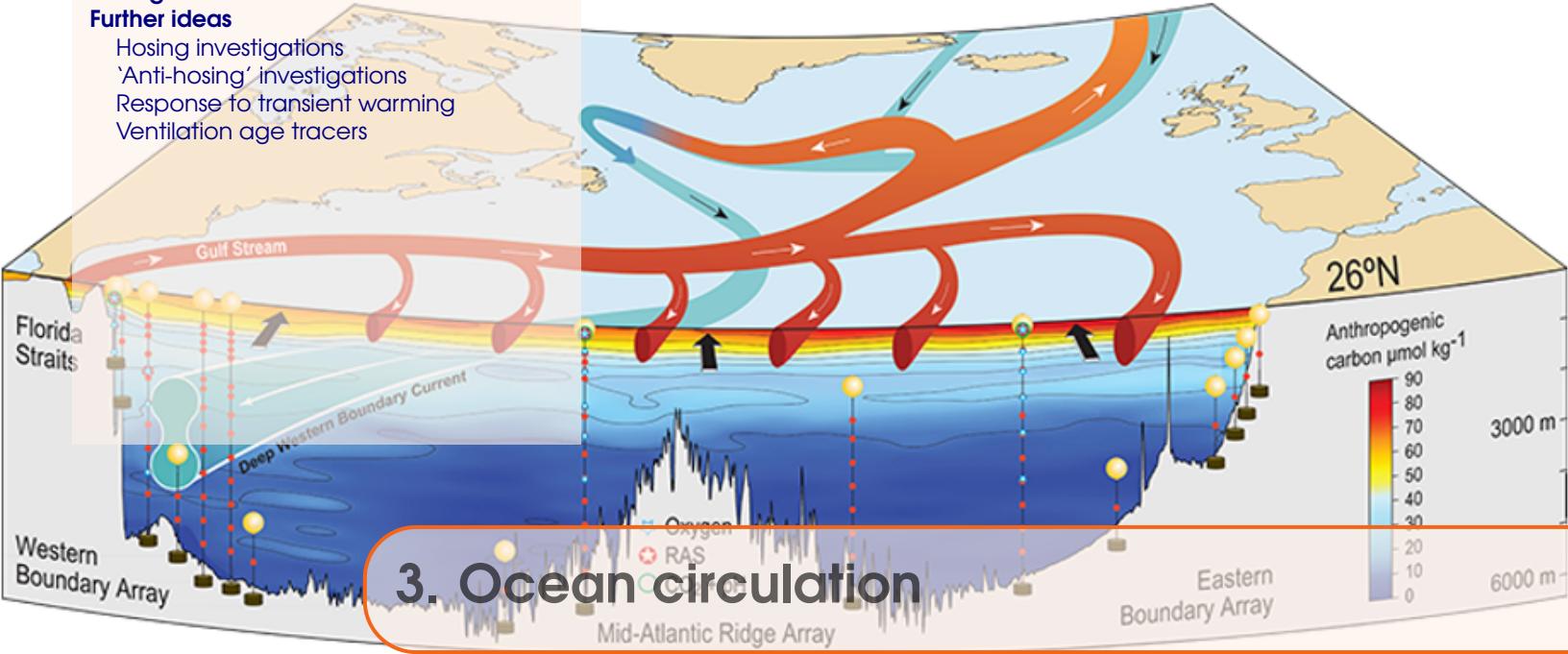
Further ideas

Hosing investigations

'Anti-hosing' investigations

Response to transient warming

Ventilation age tracers



Stuff to keep in mind:

- Nothing at all – keep your mind completely empty and let the wonderful truths of **muffin** permeate your entire being.

Background reading (Atlantic circulation and stability in **muffin**):

- Hargreaves et al. [2004] (Climate Dynamics 23, 2004, Pages 745 – 760)
→Simple assessment of the likelihood of AMOC collapse.
- Marsh et al. [2004] (Climate Dynamics, 23 2004, Pages 761 – 777)
→Characterization of thresholds of AMOC collapse.
- Singaray et al. [2008] (GRL 35, doi:10.1029/2008GL034074)
→Role of changing ocean circulation in atmospheric radiocarbon variability during the Younger Dryas.

Background reading (Miscellaneous (model) Atlantic circulation and stability):

- Rahmstorf et al. [2006] (In: Encyclopedia of Quaternary Sciences, Edited by S. A. Elias. Elsevier, Amsterdam)
→Provides the background to the Atlantic Meridional Overturning Circulation and hypothesized hysteresis.
- IPCC [2007] (e.g., Section 10.3.4)
→Future predictions of AMOC strength.
- Schmittner [2005] (Nature 434, 628– 633)
→Impacts on marine ecosystems and carbon cycling.
- Obata [2007] (J. Clim. 20, 5962–5976)
→Climate-carbon cycle model response to freshwater discharge.

READ.ME

You will need to download a new *re-start* file prior to embarking on the experiments with modern ocean circulation. To fetch this: change to the `cgenie_output` directory, and type (or copy and paste carefully from the PDF ...):

```
$ wget http://www.seao2.info/cgenie_output/LAB_2.SPIN.tar.gz
```

This downloads an archived/compressed copy of the 10,000 year *spin-up* experiment `LAB_2.SPIN`. Extract the contents of this archive by typing:

```
$ tar xfzv LAB_2.SPIN.tar.gz
```

You'll then need to change directory back to `genie-main` to run the model.

3.1 Tracing ocean circulation

The ocean biogeochemistry module (**BIOGEM**) in **muffin** provides a framework for applying time- and spatially-variable ‘forcings’ of the Earth system¹ – fluxes or restored-to boundary conditions that can be prescribed for any gas, dissolved substance (including temperature and salinity), or particulate matter. Examples include freshwater input (== a negative salinity flux forcing) of the North Atlantic to alter ocean circulation, fossil fuel CO_2 emissions to the atmosphere (== a CO_2 gas flux forcing), or aeolian iron supply to the surface ocean (a 2-D dust flux forcing).

For example: view the *user-config* file: `LAB_2.colorinjection` – you will see the following lines (under the heading: ‘# -- FORCINGS --’)

```
bg_par_forcing_name="pyyyyz_Fred"
bg_par_force_point_i=22
bg_par_force_point_j=33
bg_par_force_point_k=8
bg_par_ocn_force_scale_val_48=0.0
```

The first line points **muffin** to a directory located in `cgenie.muffin/genie-forcings` that contains a set of files that define what geochemical property is going to be altered plus information about how the magnitude of the forcing changes with time.

There are then three lines (`bg_par_force_point_i=20, ...`) that specify the location in the ocean of the geochemical forcing is going to be applied. The point sources are specified in (i,j,k) coordinates, which in this case is (22,33,08). For the ocean model resolution we are using, the grid is 36x36x16, longitude (i) is counted from left-to-right (1 to 36); latitude (j) is counted from bottom-to-top (1 to 36); level depth (k) is counted from downwards top-to-bottom (16 down to 1). Thus, (22,33,08) is a release of tracer in the North Atlantic, a little south of Greenland, and intermediate depth (level = 8 out of 16). Refer to the Figures for how the horizontal (Figure 3.1) and vertical (Figure 3.2) grid is specified.

Finally, there is a scaling parameter (`bg_par_ocn_force_scale_val_48`) which modifies the magnitude of the flux to be applied (in units of $mol\text{yr}^{-1}$).

You are going to run a brief experiment in which you will be injecting a conservative ‘dye’ tracer into the ocean. The **BIOGEM** module has two tracers defined for this purpose – ‘blue’ and ‘red’. Open the *user-config* file: `LAB_2.colorinjection` and edit the parameter controlling the flux of red dye to read:

```
bg_par_ocn_force_scale_val_48=1.0E12
```

which specifies a flux of 1.0×10^{12} ($mol\text{yr}^{-1}$) rather than zero as given as the default in the example *user-config* file².

The *base-config* you will be using is different from previously: `cgenie.eb_go_gs_ac_bg.worjh2.rb` – this specifies a 16 vertical levels ocean and also includes seasonality of solar insolation.

¹Refer to the ‘force the system’ HOW-TO in the **muffin** manual for further details on *forcings*.

²i.e. don’t leave the value as zero ... otherwise you’ll have no flux forcing applied and will not see anything happen ...

36	10	09	08	09	10	11	11	10	08	07	07	07	07	07	07	07	11	13	14	14	12	12	12	93	93	91	91	09	05	05	08	10	11	11	11	12	12	11	11
35	94	94	94	94	94	94	94	94	94	94	94	94	94	94	94	94	91	91	14	12	93	91	12	09	06	07	12	94	94	94	94	94	94	94	94	94	94		
34	94	94	94	94	94	92	92	08	07	10	15	12	09	93	93	94	94	94	91	91	91	06	05	05	06	09	12	93	93	94	94	94	94	94	94	94	94		
33	94	91	91	91	16	09	04	03	04	05	03	03	06	93	93	91	91	91	91	91	09	05	03	04	06	93	93	93	94	94	94	94	94	94	94				
32	94	91	91	91	10	04	01	01	01	01	01	01	04	11	93	91	91	91	91	91	12	05	03	03	03	11	93	93	94	94	94	94	94	94	94				
31	94	91	91	11	05	01	01	01	01	01	01	01	03	08	93	91	91	91	91	91	12	06	03	03	02	09	93	93	94	94	94	94	94	94	94				
30	91	91	91	07	02	01	01	01	01	01	01	01	02	05	93	91	91	91	91	91	09	03	01	01	03	04	02	93	10	10	94	94	94	94	94	94			
29	91	91	91	08	02	01	01	02	01	01	01	01	01	03	93	93	92	92	92	92	06	01	01	02	04	03	02	12	10	08	08	11	94	94	94	94	94		
28	91	91	16	05	01	01	01	01	01	01	01	01	01	03	10	93	92	92	92	92	05	01	01	03	03	01	03	93	12	11	94	94	94	94	92	92			
27	91	91	09	03	01	01	01	01	01	01	01	01	02	02	07	91	13	09	04	01	01	02	01	01	05	93	93	94	94	94	94	92	92	92					
26	91	91	05	02	02	01	01	01	02	01	01	01	02	05	91	08	07	04	01	01	02	01	01	09	93	93	94	94	94	91	16	09	92	92					
25	91	08	02	01	02	01	01	02	02	01	01	01	02	04	91	09	08	05	02	01	02	01	01	12	93	93	94	94	94	91	14	06	92	92					
24	93	06	02	01	02	01	01	02	01	01	01	01	02	03	06	91	10	06	03	01	03	01	02	12	93	93	94	94	94	91	09	03	92	11					
23	91	05	03	01	02	01	01	01	01	01	01	01	02	02	04	06	91	05	04	01	02	01	02	10	93	93	94	94	94	91	05	02	11	05					
22	93	06	04	01	02	01	01	01	01	01	01	01	02	02	03	04	07	91	08	03	01	01	07	93	93	94	94	94	91	03	02	09	04						
21	91	07	05	02	03	03	02	01	01	01	01	01	02	03	03	03	07	91	91	07	02	01	02	05	93	93	93	93	91	91	03	02	06	03					
20	93	03	04	02	02	03	02	01	01	01	02	02	02	03	03	03	05	91	91	11	04	02	02	03	05	08	93	93	91	11	02	02	04	03	07				
19	91	91	04	03	03	03	02	01	01	02	02	02	02	04	04	05	91	91	91	05	03	02	02	04	93	93	91	07	02	02	03	02	04						
18	93	93	07	08	07	05	03	01	01	01	02	02	02	03	04	05	91	91	91	08	04	01	02	02	03	93	93	91	04	02	02	01	02	02					
17	09	91	07	10	10	07	04	01	01	01	01	01	02	02	02	03	03	03	91	91	91	91	05	01	02	02	02	93	93	91	03	03	04	02	01	01			
16	04	93	07	14	10	05	03	01	02	01	01	02	02	03	03	02	02	91	91	91	91	05	01	03	01	02	91	91	91	03	03	04	01	01	01				
15	01	03	92	94	10	04	04	03	03	02	01	02	02	03	03	02	02	91	91	91	91	04	01	03	02	01	01	02	91	91	04	03	04	01	02	01			
14	01	02	93	91	09	04	04	04	04	01	01	03	03	03	03	02	02	91	91	91	91	03	01	03	01	01	01	02	93	91	07	03	03	02	02	01			
13	01	04	93	91	13	06	04	04	04	01	02	03	03	03	02	02	04	93	91	91	03	01	03	01	01	01	02	93	91	09	03	03	01	01	01				
12	01	08	93	91	91	08	06	04	03	01	02	02	02	03	04	02	04	93	91	91	03	01	03	01	02	03	10	08	02	03	02	01	01	01					
11	01	10	93	93	91	07	05	03	02	01	01	02	02	03	04	03	03	03	93	91	09	02	01	02	01	02	10	93	07	05	01	02	02	01	01				
10	01	05	93	92	91	05	06	03	01	01	02	02	03	04	03	03	03	93	91	06	02	01	03	02	02	07	93	05	04	01	02	03	03	02					
09	02	08	12	92	91	04	06	04	01	01	02	02	03	03	03	03	03	93	91	04	02	02	03	02	02	06	93	04	03	02	02	03	04	03					
08	01	03	03	05	92	03	05	07	01	01	01	02	02	03	03	03	04	93	09	02	02	02	03	02	01	03	05	02	03	02	02	03	03						
07	02	01	02	07	02	04	10	02	01	01	01	02	02	03	03	03	06	93	05	01	01	02	04	02	01	02	02	04	02	02	03	03	03						
06	03	02	01	01	04	02	04	12	03	02	01	01	02	04	03	02	03	08	93	03	01	01	02	03	02	01	01	03	05	02	02	04	03	03					
05	03	03	02	02	04	02	05	07	02	01	01	02	03	04	02	02	06	93	04	01	01	03	04	03	02	01	02	04	03	04	06	03	03						
04	02	02	02	03	03	04	03	01	01	02	03	03	03	02	01	02	05	05	05	04	03	02	02	04	04	03	02	02	02	01	03	05	03						
03	02	01	01	02	03	03	01	01	02	03	03	02	02	01	01	02	04	04	03	02	02	02	01	01	01	01	01	01	02	04	03	03							
02	04	10	09	09	08	07	06	03	04	03	02	02	02	02	02	03	05	09	07	04	02	01	02	03	03	04	06	06	06	08	08	04							
01	94	94	94	94	94	94	94	94	94	94	94	94	94	94	94	94	94	94	94	94	94	94	94	94	94	94	94	94	94	94	94	94							

Figure 3.2: The muffin ocean vertical level definitions for a modern 16-level ocean grid.

k	mid depth (m)	base of layer (m)
16.00	38.91	80.84
15.00	126.04	174.75
14.00	227.26	283.85
13.00	344.84	410.58
12.00	481.44	557.80
11.00	640.12	728.83
10.00	824.45	927.51
9.00	1038.59	1158.31
8.00	1287.35	1426.43
7.00	1576.33	1737.90
6.00	1912.04	2099.73
5.00	2302.02	2520.05
4.00	2755.05	3008.34
3.00	3281.33	3575.57
2.00	3892.71	4234.52
1.00	4602.92	5000.00

Run the model for ... whatever, 20 years will do. Use the *re-start* experiment that you have just downloaded to start from:

```
$ ./runmuffin.sh cgenie.eb_go_gs_ac_bg.worjh2.rb LABS  
LAB_2.colorinjection 20 LAB_2.SPIN
```

View the results – for instance how the red tracer distribution evolves with time – in the *time-slice* files (full ocean/atmosphere) properties saved in the **netCDF** format (.nc) files). You can follow the progress of the dye (and hence diagnose the properties of ocean circulation in the model) by plotting vertical and/or horizontal slices that go through (or near) the cell location in which you inject the dye tracer in the 3D **netCDF** file. Note that **Panoply** appears to ‘count’ the ocean layers in the opposite direction to the way in which the ocean model is actually counting them – the correct definition is with ‘1’ being very deepest level possible (and as displayed in the figure).

You can also view the tracer distributions in terms of a water-column integrated tracer inventory (**netCDF** variable name: ocn_int_colr; long name: colr water-column integrated tracer inventory) in the 2D **netCDF** output. (See: *Sabine et al.* [2004] for the use of water column integrals in the context of the distribution of anthropogenic CO_2 uptake and storage.) Changes in tracer inventory with time can be tracked in the time-series file

`biogem_series_ocn_colr.res.`

You can also plot the overturning circulation from the 2D netCDF file – variable phys_opsi == global overturning streamfunction, phys_opsia == overturning in the Atlantic to provide a visualization of the large-scale ocean circulation that drives tracer movement.

Spend a little while altering the flux (bg_par_ocn_force_scale_val_48) and/or location (bg_par_force_point_i, bg_par_force_point_j, bg_par_force_point_k) of tracer input. Overall – note how you can use numerical ‘tracers’ to help diagnose (and better understand) the circulation of the ocean.

An interesting (honest!) and illustrative exercise is to use the dye tracer to pick out the path taken by Mediterranean Intermediate Water. Despite the low resolution of the **muffin** ocean circulation model component and the highly restricted representation of the Mediterranean, the model does predict a salty Mediterranean as a consequence of P-E in this basin (and its catchments) being negative and this water makes its way out in the subsurface into the Atlantic.

Simply specify a dye injection somewhere in the Mediterranean (be careful with the restricted depth of the Mediterranean – if you inject too deeply (into the crust!) then you will not see anything (refer to the figure for the depth level (k) number of the maximum depth of the water column in each location), and it is better to inject it relatively close to the opening of the gateway (try some different locations and see which ones produce a reasonably instructive tracing of Mediterranean outflow). Run for e.g., 20 or 50 years (from the provided spin-up). Then:

1. View the dye-tagged plume of Mediterranean Intermediate Water by plotting a lat-lon slice (from the 3D **netCDF** file). This will give you the depth of the plume. How does this compare with salinity observations (salinity observations and appropriate global datasets can be found on the web with a little patience)? You can also view the water-column integrated distribution (2D **netCDF**).

2. Try viewing the plume via a lat-depth slice. Refer to the figure to determine the ‘i’ value up the Atlantic that will just graze the edge of what passes for Spain at this low model resolution. Which direction does it head after exiting the Mediterranean? Is this ‘realistic’?

3.2 Poking the climate beast

Instead of adding a dye tracer, you could add fresh water to the ocean surface to assess the sensitivity of the Atlantic Meridional Overturning Circulation (AMOC) to collapse, in a classic ‘hosing’ experiment.

The *user-config* file for this is called: LAB.2.hosing. The default (i,j) location of the flux input is the same (as the dye tracer), but now the injection at the surface (level: k=16). Note that the forcing of the salinity tracer is negative (freshwater = negative salinity compared to sea-water)!

To orientate you in freshwater forcing space: bg_par_ocn_force_scale_val_2=-2.0E17 should be sufficient to make ‘stuff happen’ and quickly. BUT, this is a pretty extreme flux (see overleaf for a rough conversion between salinity forcing units (mol yr^{-1}) and fresh water flux (in $\text{m}^3 \text{ s}^{-1}$ or Sv). Much more than this and the model may crash or at the very least, you’ll be left with a large freshwater pond in the North Atlantic ... (See later (Section 1.6e) for some exciting discussion on units!)

To run the model for e.g., 20 years using the same restart:

```
$ ./runmuffin.sh cgenie.eb_go_gs_ac_bg.worjh2.rb LABS
LAB_2.hosing 20 LAB_2.SPIN
```

20 years should be long enough to see a collapse start to occur, but you might want to run the model for longer (and it can be submitted as a job, of course). Running for longer will also allow you to have a smaller, less extreme (and maybe more realistic) freshwater input flux.

The most obvious property of the Earth system to follow is the Atlantic overturning strength (*biogem_series_misc_opsi.res*). The AMOC stream-function (in *fields_biogem_2d.nc* 2-D time-slice **netCDF** results file, field: *phys_opsia*) is also illustrative. You can also try and identify the salinity anomaly (see below) due to freshwater input in the 3D salinity tracer field.

There are also important impacts on surface air temperatures and maybe sea-ice extent (in *fields_biogem_2d.nc*). Note the importance (sort of) of the AMOC in transporting heat to the N Atlantic region (the film the Day After Tomorrow was not entirely inaccurate in this particular respect). Be aware of the possibility of climate impacts far from the location of fresh water forcing. Look out for any significant-looking impacts on sea-ice extent, etc.

Note that as the model is running rather slow than in the snowball configuration, you might want to think carefully of making use of cluster queuing possibilities (i.e., running multiple experiments at once in the background).

To more easily assess some of these impacts (and for other sorts of analysis) it is possible to create an anomaly (difference) map in **Panoply**:

1. First open a dataset, e.g., *atm_temp* (surface air temperature) in the 2D **netCDF** file. You can either double-click the variable name, or, with the variable name highlighted, click the ‘Create Plot’ icon.
2. Now, with the *atm_temp* still selected (and the first plot window still open), click on the ‘Combine Plot’ icon. A dialogue box will appear and ask you to select a plot to combine the new one with. Make sure the name of your first plot window is selected/highlighted. Click ‘Combine’. OR, simply drag a second dataset into the plot window of the first dataset.

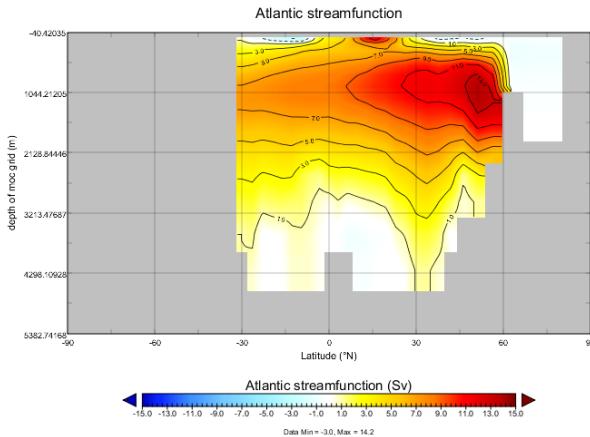


Figure 3.3: Example plot of (normal/default modern) overturning streamfunction (2D **netCDF** file). (e.g., for Atlantic: **netCDF** parameter name: `phys_opsia`, long-name: Atlantic streamfunction). Note that autoscaling has been turned off and the min and max plotting limits set manually. By convention, streamfunctions are plotted with their scale symmetrical around zero, giving red and ‘warm’ colors for positive value and clockwise overturning, and blues and ‘cold’ colors for negative values and anti-clockwise overturning. (The plot has been tart-ed up by overlaying solid contours plus contour labels.) It may be necessary in **Panoply** to re-orient (invert) the vertical grid.

3. You now have a plot window that by default it is showing you the difference between two identical (in time) slices. The two different slices are labeled Array 1 (LH side) and Array 2 (RH side).

Keep one array (Array 1) fixed to the initial (year 1 (centered on 0.5)) and vary the year in the second array (Array 2). Note that you can select in Panoply whether Array 1 – Array 2 is plotted, or Array 2 – Array 1, or various proportional or relative differences.

Note that you can switch off the auto-scaling feature (Always fit to data) and center the scale so that no change is white, with positive deviations = red and negative = blue by clicking on Center on 0 (an often used convention in climate field plotting).

Two example plots (using **Panoply**) are shown in Figure 3.3 for the Atlantic basin, and Figure 3.4 for the Pacific.

You can also plot ocean current fields which is sort-of fun and maybe even informative(!):

1. In the 3D **netCDF** file, the three components of ocean velocity are represented by the variables: ocean velocity – u (Eastwards), ocean velocity – v (Northwards), and ocean velocity – w (upwards). 2. Open up velocity – u. Chose ‘lon-lat’.
2. Select/highlight velocity – v. and click on the ‘Combine Plot’ icon (as per before).
3. Rather than a difference map, which is what you get by default, i.e., ‘Array 1 – Array 2’ – from the drop-down menu (next to the ‘Interpolate’ button) select ‘Vector Magnitude’.
4. You should have a color contoured (or not if you prefer plotting without contouring on) map of ocean current speed, with velocity vectors (direction and magnitude) overlain. You’ll need

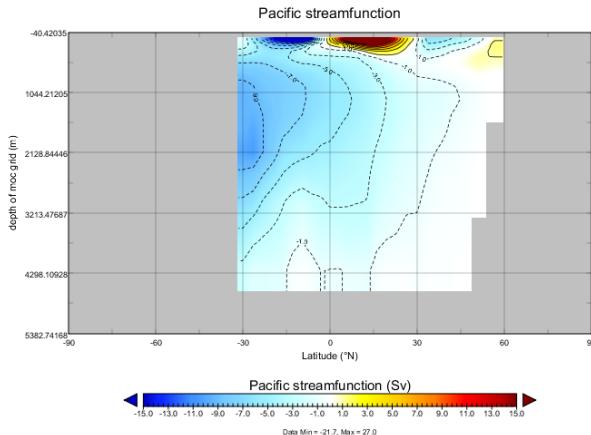


Figure 3.4: Example plot of collapsed AMOC.

to re-scale the velocity vectors to properly see them – from the ‘Contours and Vectors’ tab – change the ‘Scale Length’ to e.g., 0.1. (On a Mac, look under the ‘Vectors’ tab for a ‘Reference Value’ to something like 0.1.) When fresh-water hosing – look out for impacts on the N. Atlantic current system associated with the AMOC.

5. You can repeat this for deeper depth levels in the ocean – e.g., between about 1500 and 2000 m is a good place to go looking for the Western boundary current (and AMOC return) in the model (such as it exists at this low resolution) but you’ll need to re-scale the velocity vectors again (e.g., to 0.01 to less).

An example plot (using **Panoply** for visualizing surface ocean current fields) is given in Figure 3.5.

Finally, a brief note on units ... the freshwater forcing is implemented as negative salinity, just to really screw with your mind. The generic internal **muffin** model units for the forcing end up as $PSU\text{kg}^{-1}\text{yr}^{-1}$. Which sort of does not make much sense ...

Start, by thinking of a value of `bg_par_ocn_force_scale_val_2` of -34.9 as equivalent to taking all the salt out of 1kg of freshwater (since the mean global salinity is 34.9PSU). Or equivalently, since the ocean volume is fixed, an applied forcing value of -34.9 is equivalent to adding 1kg of freshwater to a (surface) box. So, a value of `bg_par_ocn_force_scale_val_2` of -3.49×10^4 ($-3.49E04$) would be a flux of $1\text{m}^3\text{yr}^{-1}$ (1000kgm^{-3}) of freshwater.

So, in the example earlier (`bg_par_ocn_force_scale_val_2=-1.0E18`), the freshwater flux is $1.0 \times 10^{18} / 3.49 \times 10^4 = 2.8653 \times 10^{13} \text{m}^3\text{yr}^{-1}$.

The literature invariably gives freshwater fluxes in units of Sv ($10^6 \text{m}^3\text{s}^{-1}$). So in the example, the freshwater flux is: $9.0797 \times 10^5 \text{m}^3\text{s}^{-1}$ ($365.25 \times 24 \times 3600 = 31557600\text{syr}^{-1}$). Or 0.9Sv . Read the literature ... but generally, fluxes of ca. 0.05Sv and larger (and to quite specific places) are applied in models to induce an AMOC collapse.

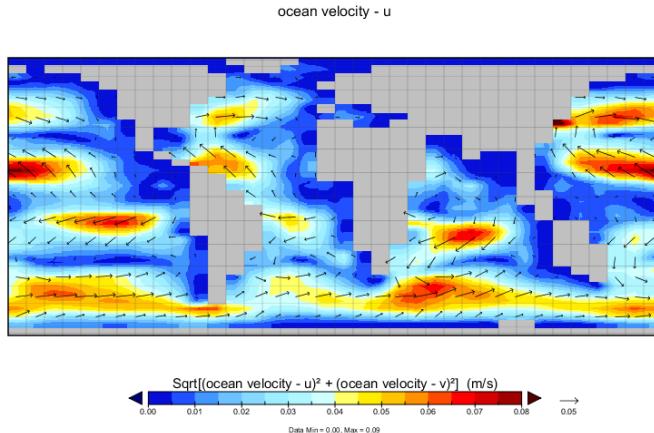


Figure 3.5: Example plot of (normal/default modern) ocean current fields (3D netCDF file). Again scaling has been set manually to create an easy-to-interpret axis scale. On the left is the surface field, and on the right an intermediate depth (illustrating what approximates the Deep Western Boundary current in the model in the Atlantic).

3.3 Further ideas

3.3.1 Hosing investigations

What is the largest freshwater flux that can be sustained without ‘collapsing’ the AMOC? Is there a ‘threshold’ (‘tipping point’) of freshwater input, beyond which the AMOC rapidly decreases in strength?

Is the precise location of the freshwater input important (i.e., try tipping it in somewhere else)? What would you expect to see in the paleo (e.g., ice core) record of both hemispheres if such a shutdown occurred in the past?

Are any other major regions of deep water formation (where are they) sensitive to freshwater perturbation and what are the consequences (could it happen in the future)?

3.3.2 ‘Anti-hosing’ investigations

There are questions concerning past changes in the AMOC as to whether it is ‘pushed’ or ‘pulled’. i.e., if the AMOC shoals in depth and/or weakens, is it because its production has weakened, or as Antarctic Bottom water (AABW) strengthened and ‘pushed’ it out of the way (to shallower depths)?

What you might try then is to inject salt in the Southern Ocean as opposed to fresh water in the North Atlantic. All you need do is pick an appropriate grid point (this is worth thinking about carefully and maybe testing different locations) and rather than giving the parameter `bg_par_ocn_force_scale_val_2` a negative value, you give it a positive one. (Start by trying similar magnitudes of value as before and see what happens.)

Is the AMOC (for the same magnitude of forcing) more sensitive to being ‘pushed’ or ‘pulled’? (Obviously the answer will very much depend on where the perturbations are being applied.)

3.3.3 Response to transient warming

A current concern regarding anthropogenic climate change is the ocean circulation (and marine ecological and biogeochemical) response to a strong warming of the surface, as rapid surface warming is assumed (and demonstrated in models) to result in surface stratification of the ocean, likely restricting the nutrient supply to phytoplankton and reducing ventilation of the ocean interior with dissolved oxygen.

You can explore the transient response of ocean circulation to warming by simply adjusting the radiative forcing parameter as per used in the snowball Earth experiments: `ea_radfor_scl_co2`. By default in the modern continental configuration, this has a value of 1.0, corresponding to 278 ppm atmospheric CO_2 . A value of 2.0 would reflect warming equivalent to 556 ppm CO_2 . And 3.0 more like an end-of-the-century warming. Note that you are applying the warming instantaneously by manipulating the climate system in this way and hence the changes will be more extreme than those occurring over the time-scale of this century. Also note that a cooling could be applied instead. A *user-config* – `LAB_2.EXAMPLE` – is provided as a template for these experiments.

Potentially interesting properties of the Earth system to look at include sea-ice extent and AMOC strength (in the ASCII time-series files), and the overturning streamfunction and sea-ice extent in the 2-D **netCDF** output.

How much radiative forcing is required to collapse the AMOC? What atmospheric CO_2 value does this approximately correspond to?

3.3.4 Ventilation age tracers

muffin has the capability to employ/simulate a ventilation age tracer³ – a numerical tracer in the ocean that tracks the time since a parcel of water last ‘saw’ the surface. To do this, requires both ‘red’ and ‘blue’ numerical tracers to be selected. **muffin** can then be directly apply the forcings necessary to each tracer in order to simulate water mass ventilation age.

Provided is an additional *user-config* file: `muffin.worjh2.NONE_colage.EXAMPLE`.

A new *re-start* experiment is provided called `muffin.worjh2.NONE_colage.SPIN` and which can be downloaded as per before:

```
$ wget http://www.seao2.info/cgenie_output/muffin.worjh2.NONE_colage.SPIN.tar.gz
```

and unpacked, as before, by:

```
$ tar xfzv muffin.worjh2.NONE_colage.SPIN.tar.gz
```

(Remember: you should be in the `cgenie_output` directory when you do this.)

To run (e.g. for 10 years), following on from its *re-start* experiment:

```
$ ./runmuffin.sh cgenie.eb_go_gs_ac_bg.worjh2.rb LABS muffin.worjh2.NONE_colage.EXAMPLE
10 muffin.worjh2.NONE_colage.SPIN
```

(all on one line)

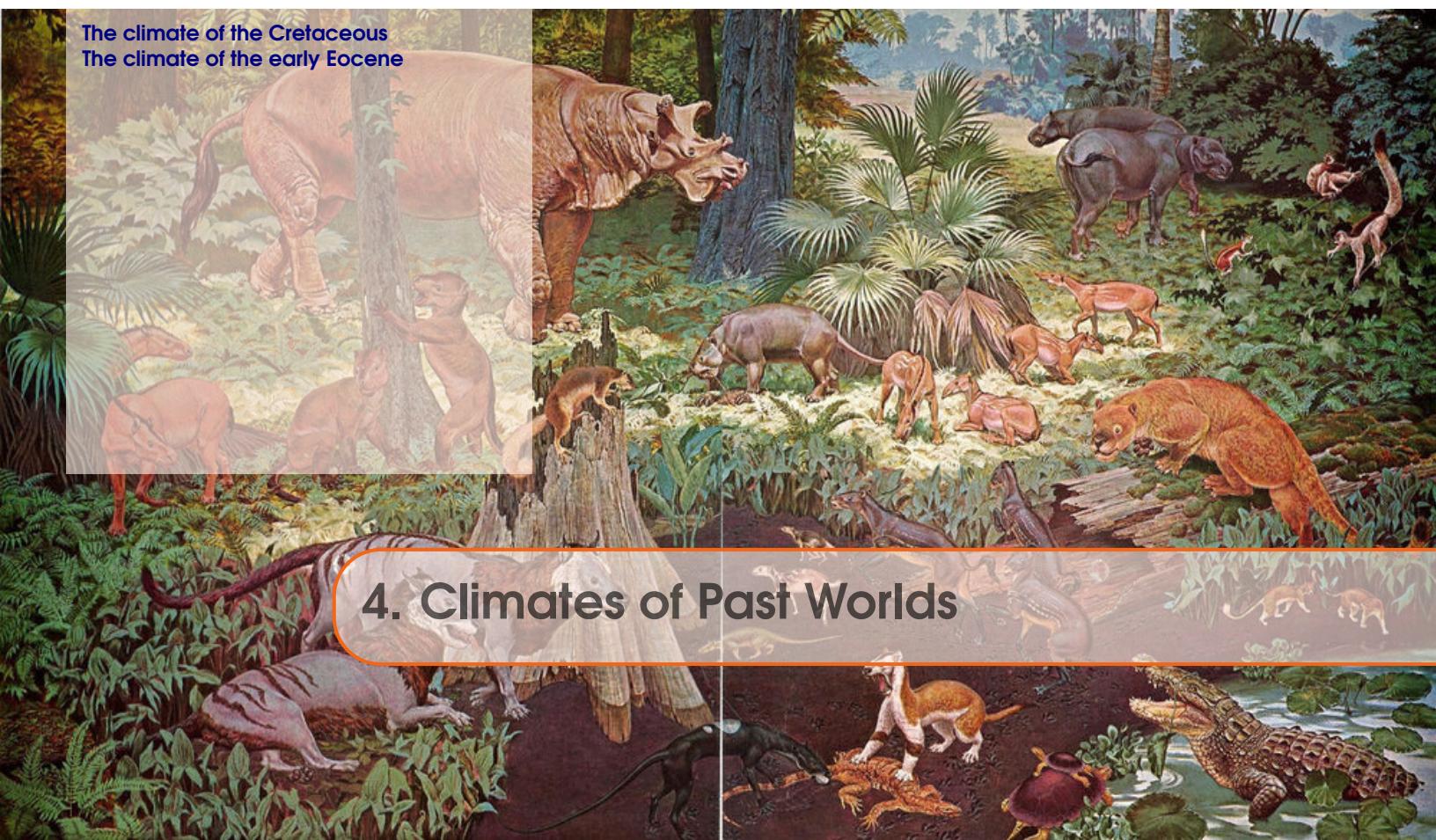
In the 3D netCDF output file – `misc_col_Dage` is the output variable that is the calculated ventilation age.

First – explore the distribution of water mass age and think about the physical ocean circulations reasons for this. Are all the modelled distributions reasonable? Are there indicators of facets of the

³Under the ‘screw with and/or diagnose the climate system’ HOW-DO – see ‘Add a water mass age tracer’ (and the ‘easy’/automatic method described towards the end of that section).

simulated circulation that are not particularly realistic? Try plotting both lon-lat and lat-depth slices through the ocean.

Secondly – you could additionally combine this experiment configuration with fresh water hosing, or a prescribed radiative forcing (and warming), and explore how ocean ventilation responds as ocean circulation changes. (You'll need to create for yourself a new *forcing* directory and set of files that now combines both color tracers AND a freshwater flux.)



4. Climates of Past Worlds

This Chapter comprises 2 different exercises, both employing 'realistic' reconstructions of past continental configuration and climate and designed to explore some of the key controls on global surface climate as well as the spatial pattern of surface temperatures.

The two exercises concern the late Cretaceous and early Eocene climate states, and are somewhat interchangeable.

Background reading (Cretaceous climate)

- Barron, E.: A warm equable Cretaceous: the nature of the problem, *Earth-Science Reviews* 19, 305–338, 1983.
- Bice, K. L., and R. D. Norris, Possible Atmospheric CO_2 extremes of the middle Cretaceous (late Albian-Turonian), *Paleoceanography* 17, doi: 10.1029/2002PA000778, 2002.
- Bice, K. L., B. T. Huber, and R. D. Norris, Extreme polar warmth during the Cretaceous greenhouse?: Paradox of the Late Turonian $d^{18}O$ record at DSDP Site 511, *Paleoceanography* 18, doi: 10.1029/2002PA000848, 2003.
- Bice, K. L., D. Birgel, P. A. Meyers, K. A. Dahl, K. Hinrichs, and R. D. Norris, A multiple proxy and model study of Cretaceous upper ocean temperatures and atmospheric CO_2 concentrations, *Paleoceanography* 21, PA2002, doi:10.1029/2005PA001203, 2006.
- Donnadieu, Y., et al., Modelling the primary control of paleogeography on Cretaceous climate, *Earth and Planetary Science Letters* 248, 426–437, 2006.
- Huber, B. T., Norris, R. D., and MacLeod, K. G.: Deep-sea paleotemperature record of extreme warmth during the Cretaceous, *Geology* 30, 123–126, 2002.
- Hunter, S. J., et al., Modelling Maastrichtian climate: investigating the role of geography,

- atmospheric CO_2 and vegetation, *Clim. Past Discuss.* 4, 981–1019, 2008.
- Jenkyns, H. C., Forster, A., Schouten, S., and Sinninghe Damste, J. S.: High temperatures in the Late Cretaceous Arctic Ocean, *Nature* 432, 888–892, 2004.

Background reading (Eocene climate)

- Dunkley Jones, T., D.L. Lunt, D.N. Schmidt, A. Ridgwell, A. Sluijs, P.J. Valdes, and M. Maslin, Climate model and proxy data constraints on ocean warming across the Paleocene-Eocene Thermal Maximum, *Earth-Science Reviews* 125 123–145 (2013).
- Lunt, D. J., Dunkley Jones, T., Heinemann, M., Huber, M., LeGrande, A., Winguth, A., Loptson, C., Marotzke, J., Roberts, C. D., Tindall, J., Valdes, P., and Winguth, C.: A model–data comparison for a multi-model ensemble of early Eocene atmosphere–ocean simulations: EoMIP, *Clim. Past*, 8, 1717–1736, doi:10.5194/cp-8-1717-2012, 2012.

READ.ME

You will need to download new *restart* files for this Chapter (both 10,000 year spin-ups). These are:

Cretaceous: \$ wget http://www.seao2.info/cgenie/labs/AWI.2017/LAB_5.SPIN.tar.gz

Eocene: \$ wget http://www.seao2.info/cgenie/labs/AWI.2017/LAB_5b.SPIN.tar.gz

Extract the results in the usual way and in the usual place ... and return to `genie-main`.

4.1 The climate of the Cretaceous

A previously spun-up state of Maastrichtian climate with **cGENIE.muffin** in a ca. 70 Ma configuration¹ (LAB_5.SPIN) is provided as a starting point. A *user-config* (LAB_5.EXAMPLE) that continues on this climate state is run:

```
$ ./runmuffin.sh cgenie.eb_go_gs_ac_bg.p0067f.NONE LABS
LAB_5.EXAMPLE 10 LAB_5.SPIN
```

Your task now is ... ‘simple’: Account for the Cretaceous reduced Equator-to-pole surface temperature gradient (compared to modern), particularly the apparently much warmer poles. Different data-based time-slices (including the Maastrichtian) are provided in *Huber et al.* [2002], although the low latitude $\delta^{18}\text{O}$ based temperatures are now not considered reliable. *Jenkyns et al.* [2004] contains high latitude (Arctic) data for the Maastrichtian. There are proxy-derived latitudinal temperature gradients and model-data studies for earlier in the Cretaceous – the problem is essentially the same.

The 2-D NetCDF results file contains the surface air temperature field (and sea-ice cover, if any). The 3-D NetCDF results file contains fields for ocean temperatures (and salinity). Both contain continental configuration and ocean bathymetry.

Panoply will plot the zonal average for you (as used in model-data comparisons – e.g., see Bice and Norris [2002]) – in the Array(s) tab, the Plot can be set to Zonal Averages rather than Map. You can get a smooth curve by selecting Interpolate. Remember you can set (and fix) scales rather than let Panoply auto-scale continually.

The following ‘controls’ over the climate system are provided to you in the form of a list of parameters at the bottom of the LAB_5.EXAMPLE *user-config* file for editing:

```
# --- MISC -----
#
# === ATMOSPHERE ===
# \(\text{CO}_2\)\ radiative forcing scaling factor [DEFAULT = 4.0]
ea_radfor_scl_co2=4.0
# CH4 radiative forcing scaling factor [DEFAULT = 1.0]
ea_radfor_scl_ch4=1.0
# Equator-to-pole different in planetary albedo [DEFAULT = 0.260]
ea_albedop_amp=0.260
# Baseline planetary albedo [DEFAULT = 0.200]
ea_albedop_offs=0.200
# atmospheric diffusivity of temperature (horizontal) [DEFAULT = 5.0e6]
ea_12=5.0e6
# === OCEAN ===
# ocean diffusivity of temperature + salinity (horizontal) [DEFAULT = 1494.4]
go_14=1494.4
# scaling for wind stress (set values of both identical) [DEFAULT = 1.531]
go_13=1.531
ea_11=1.531
```

Most of these parameters are associated with the radiative forcing of climate or atmospheric transports. The most useful ones are likely to be:

¹Note that for speed, no carbon cycle is selected in this configuration.

(i) The line:

```
ea_radfor_scl_co2=4.0
```

which specifies a radiative forcing of climate by CO_2 equivalent to 4 times modern CO_2 (i.e., $4 \times 278ppm = 1112ppm$) as per before (e.g. snowball Earth experiments).

The line:

```
ea_radfor_scl_ch4=1.0
```

specifies a radiative forcing of climate by CH_4 equivalent to 1 times modern CH_4 (i.e., $1 \times 700ppb$).²

- (ii) cGENIE, as configured here, does not have a land surface scheme (no snow cover) nor clouds nor ice sheets, so a planetary albedo is prescribed (see Figure 4.1). This varies with latitude and is parameterized after a fully coupled GCM simulation. There is a parameter which controls how the albedo varies as a function of latitude, which can be adjusted:

```
ea_albedop_amp=0.260
```

However, if you vary this, why are you doing it? (What is the physical justification for giving the poles a higher or lower albedo relative to the Equator?) There is also a parameter that sets the baseline (minimum) albedo:

```
ea_albedop_offs=0.200
```

i.e., albedo is primarily a sum of the baseline value plus the Equator-to-pole slope (times the latitude).

- (iii) The diffusivity of heat in the atmosphere (since it is a simple 2-D atmospheric model, with no atmospheric circulation, the atmosphere is made ‘diffusive’ to help capture heat and moisture transport) is:

```
ea_12=5.0e6
```

Note the maximum value the model can cope with is ca. $1.0E7$.

Run the model for however long you think is ‘necessary’ (/justified). The surface climate will approach equilibrium ‘relatively’ quickly. Deep ocean temperatures will typically take thousands of years to fully adjust ... You can assess how the model approaches equilibrium most easily from the atmospheric temperature time-series results file, and from the ocean temperature time-series results file (allowing to you to contrast surface and whole ocean temperature changes).

You can also try tracing and analyzing the patterns of ocean circulation in the Cretaceous world, in the same way as you did for the modern climate system. And in fact, it would be a useful exercise to directly compare your previous modern results vs. Cretaceous results, or even better, carry out a set of paired experiments (e.g., similar locations and fluxes of dye injection) for both modern and Cretaceous. (Note that if you do this and swap back and forth between modern and Cretaceous configurations, you cannot submit straight away to the cluster but must first briefly run

²But effectively, this is going to do exactly the same as changing radiative forcing due to CO_2 .

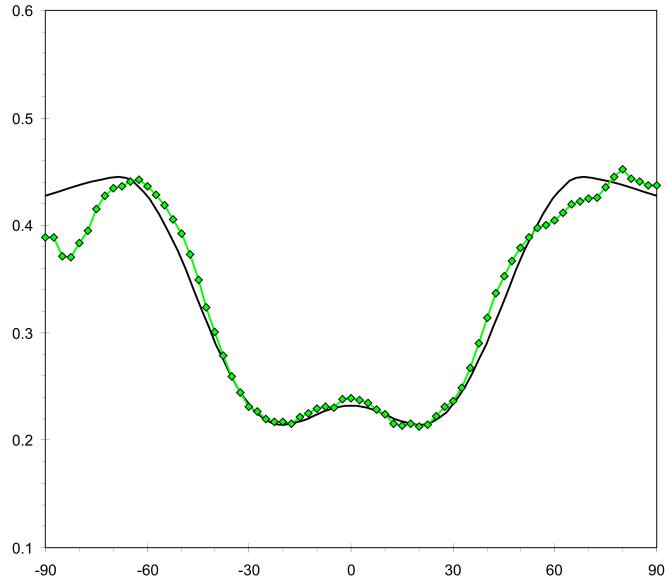


Figure 4.1: Prescribed planetary albedo. The latitudinal (from 90°S (-90°N) on the left, to 90°N on the right) profile of planetary albedo as calculated in a fully coupled GCM is given in green, and the cGENIE ‘fit’ in black.

the experiment at the command line in order that cGENIE is compiled into the new continental configuration.)

A new *user-config* (LAB_5.colorinjection) that continues on from this climate state, can be run:

```
$ ./runmuffin.sh cgenie.eb_go_gs_ac_bg.p0067f.rb LABS
LAB_5.colorinjection 10 LAB_5.SPIN
```

(Also note that the *base-config* is different as it now includes the definition of 2 dye tracers.)

The default locations for the dye release are set different compared to in the modern configuration. BUT, the default location is not necessarily ideal / particularly revealing ... so you’ll need to look count grid cells West-to-East (the i direction) and from South to North (the j direction) in order to determine a more suitable location for tracing circulation.

By means of dye tracing, looking at the (global only) overturning stream-function, and/or temperature and salinity (and density) profiles, see if you can identify where in the Cretaceous ocean deep water forms in the model. AND, more importantly, think about WHY does it form where it does?

Also: what about surface ocean circulation and gyres? Are these located where you would expect (e.g. based on your understanding of modern ocean circulation). Plot velocity vectors to help, and/or refer to the Barotropic streamfunction output in the 2D netCDF file.

4.2 The climate of the early Eocene

A previously spun-up state of early Eocene climate with cGENIE in a ca. 56 Ma configuration (LAB_5b.SPIN) is provided as a starting point. A user-config (LAB_5b.EXAMPLE) that continues on this climate state is run:

```
$ ./runmuffin.sh cgenie.eb_go_gs_ac_bg.p0055c.NONE LABS  
LAB_5b.EXAMPLE 10 LAB_5b.SPIN
```

(Note that once again, for speed, no carbon cycle is selected in this configuration.) Again, as per for the late Cretaceous configuration, a configuration that includes the ocean dye (color) tracers is also provided. The corresponding user-config (LAB_5b.colorinjection) that continues on this climate state can be run as follows:

```
$ ./runmuffin.sh cgenie.eb_go_gs_ac_bg.p0055c.rb LABS  
LAB_5b.colorinjection 10 LAB_5b.SPIN
```

What to ‘do’ with this? Again – start by tracing and analyzing the patterns of ocean circulation in the Eocene world, as per how you did for the modern and Cretaceous system. Where does deep water form in the Eocene? Is the surface ocean circulation as expected? Can you see any patterns emerging? At least in model world – what seems to be dictating the locations of deep water formation? Does overturning and/or surface circulation appear to significantly vary as a function of continental configuration or radiative forcing? If not, why not?

But how do you know how close, or not, the model climate is to ‘reality’? Available from the website, on the LH side of the page under ‘got data?’ at the bottom (item (7)), are 2 netCDF files containing proxy sea surface temperature (SST) data re-gridded to the cGENIE model grid. One is for pre-PETM SSTs, and one for peak PETM SSTs. (The data are from Dunkley Jones et al. [2013].)

These netCDF files can be opened and visualized in exactly the same way in exactly the same way as per the cGENIE output files. You’ll see a few (there are not many!) data points, in a sea of grey (which stands for ‘no value’). (Note that it is best to turn data interpolation OFF.)

More useful . . . is that the data distribution can be combined with the model ocean temperature output field, in a difference map (see earlier for notes on creating difference maps). You can thus visualize the pattern of model-data mismatch. (Note you might want to hit ‘Fit to data’ to autoscale the plot, or simply pick and manually enter a +/- limit for the data plotting, perhaps ensuring it is symmetrical about zero such that red anomalies could represent proxy SST values higher than in the model, and blue point data being lower than in the model.)

Furthermore, in the Arrays tab, you can switch to a Zonal Average view rather than a Map view (...it is probably easier to visualize the model-data misfit in this way.)

So, starting with the pre-PETM data and given model configuration, make some assessment of how the model fits the data (or not). What might be the reason for the misfit? You might test adding, and adjusting, some of the parameter values controlling surface SST from the Cretaceous climate exercise. Can you reduce the model-data misfit?

Finally, the PETM SST data given in the 2nd netCDF file are from the peak of the PETM. The ocean is rather warmer as compared to just prior to the PETM. Your task is to determine (either using the default model configuration, or your adjusted configuration) much radiative forcing (parameter `ea_radfor_scl_co2`) and hence CO_2 is required to explain this warming?

Generating fake Worlds

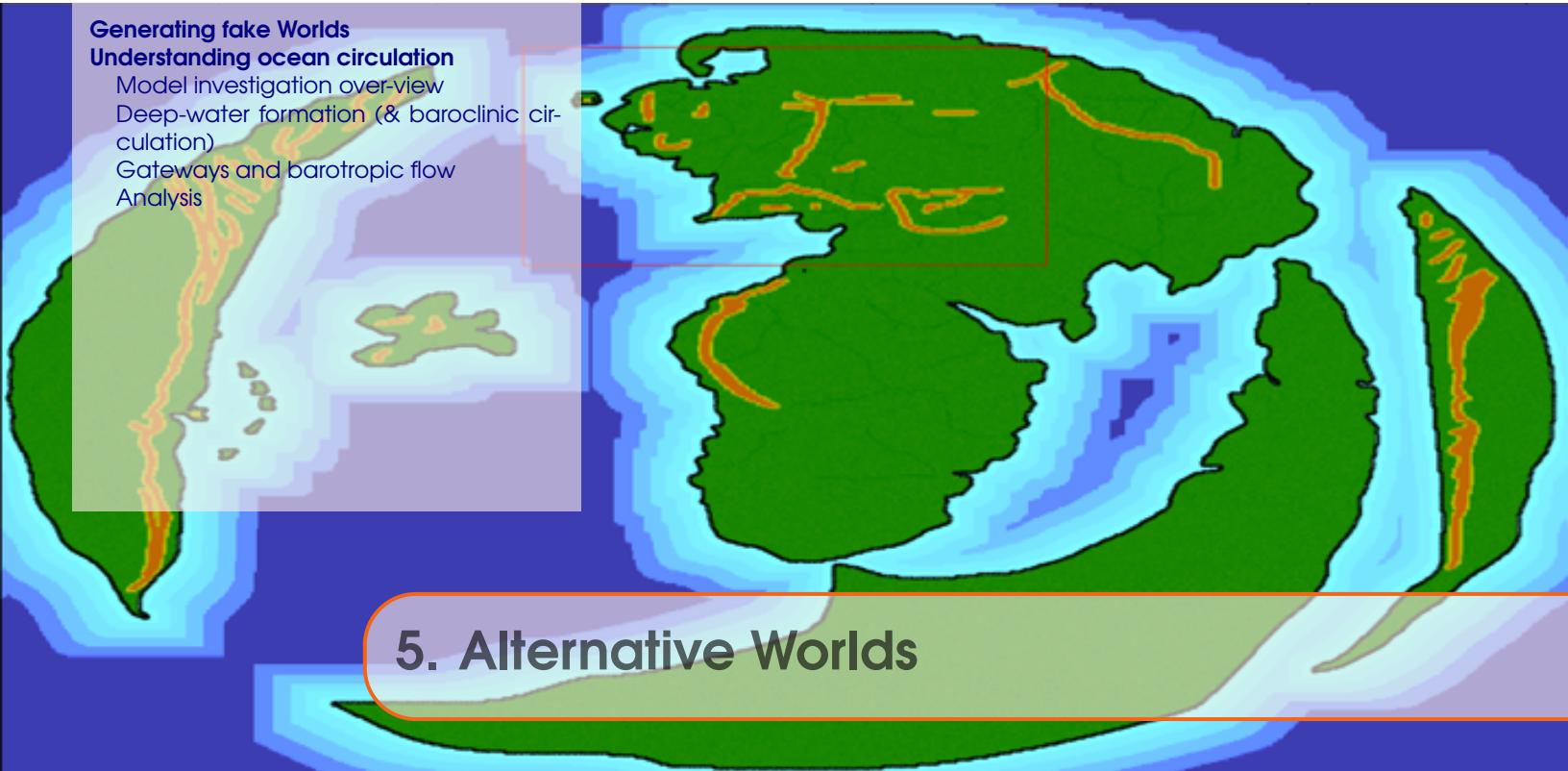
Understanding ocean circulation

Model investigation over-view

Deep-water formation (& baroclinic circulation)

Gateways and barotropic flow

Analysis



5. Alternative Worlds

Fake (alternative) Worlds are a useful tool for developing a general understanding of how climate (and ocean circulation) and global biogeochemical cycles operate. Different model resolutions and continental configurations can be generate and run to test specific hypotheses about the role and sensitivity of different elements in Earth system to emergent patterns of climate (and biogeochemical cycling). Ensembles of differing Worlds can also be utilized to more generalize understanding as well as in a fishing expedition of new or unexpected phenomena and system behaviours.

This Chapter will take you through the generation and exploration (analysis) of fake (alternative) Worlds.

5.1 Generating fake Worlds

TO COME ...

5.2 Understanding ocean circulation

What follows, outlines different ways of visualizing and quantifying ocean circulation in **muffin**, and coming to a model-empirical understanding of ocean circulation controls.

5.2.1 Model investigation over-view

In restricting model-based experiments and derived understanding to the modern continental configuration (and climate) in **muffin**, you'd be unlikely to come to any particularly deep or fundamental insights into ocean circulation controls in general. In any case, the modern and future (and recent glacial past) ocean circulation states and dynamics has been picked at endlessly by 1000s of researchers, writing 100,000s of publications ... and yet still we do not fully understand modern ocean circulation, let along that at the last glacial.

A better (more fun) approach is to generalise the problem and consider fundamentally different configurations of ocean basins and climate. This is what you will be doing. But first, some potted advice:

- Try and define a hypothesis, or hypotheses, to pursue and test. These need only be very rough at the outset – you'll likely find that you get new ideas and can either refine your original hypotheses, or come up with new ones, as you start to play with the model and see what is feasible and not feasible in terms of analysis.
- Create a plan of study – what sort of experiments, how many, how are they going to be analysed, how do they all fit together in addressing the overarching hypotheses? Again – it is very likely that your plan will evolve, but try and start out with something to guide you rather than wandering randomly in model world ...
- For creating and using different worlds – create a list or table of the configurations to be explored, and summarize what you are finding. It might be helpful to plan out on paper first, some of the main continental configurations you need to create and then test in the model. (Again, this will evolve.)
- Once you have some sort of idea what are going to do – plan the work. This is important because to run the model to steady state is not trivial. Perhaps plan on having to run the model for 5,000 years¹ in order to achieve a fully spun up ocean circulation state. You might get away with shorter runs, but know in advance what sort of error this would induce and is it 'important'? Once you have a list and know how long each might take, you can plan out when they will be run on the cluster (presumably on the queue) and when the results will be analysed. Note that it is a virtual certainty that once you have analysed the results of the first set of experiments, you will want something different (and will then need to revise the plan and list of experiments and analysis etc.).

Overall – scientific investigations with simplified (and relatively fast) Earth system models tends to become a somewhat iterative undertaking and can involve significant trial-and-error. (This is healthy and to be expected and even enjoyed!) Rarely, can you devise at the outset, a single run or set of experiments, and it turns out this is completely sufficient. If you do not see something you do not understand in the results of the initial model experiments, you are either God (e.g. the Spaghetti Monster or Invisible Pink Unicorn), or not doing it properly. Expect to see things that interest you

¹Even so, you should try running the model much longer to be confident that 5,000 years is OK (and sufficiently close to final equilibrium).

and lead you off on a tangent (hopefully – this is how research should be).

Finally – note that in swapping between different continental configurations, **muffin** currently requires the model executable to be re-compiled.² This ... cannot happen on the compute nodes of the cluster if you submit an experiment using a different *base-config*. When changing to a new *base-config*: first, run the model briefly interactively (i.e. at the command line). Once it has compiled (and started running), the experiment can be killed (**Ctrl-C**) and now it is good to submit to the queue as a job. In any case, in utilizing a new configuration that you may never have used before, it is good practice to test it first (e.g. watch it run for a short period).

5.2.2 Deep-water formation (& baroclinic circulation)

The first investigation is to investigate the controls on the strength and large-scale structure of ocean circulation, particularly in terms of the global meridional overturning circulation (MOC), and associated with this – the primary sites of deep-water formation.

There are presumably 2 key controls to this:

1. Temperature.

This is presumably mostly a function of latitude (i.e. the further North or South, the more suitable the site will tend to be for deep-water formation) and to some extent season. There will also be an influence of the prescribed planetary albedo (which includes the effect of clouds) as well as of sea-ice (if it exists).

2. Salinity.

This will generally be controlled by P-minus-E – the balance between precipitation to the ocean surface as well as fresh-water run-off from the continents, vs. evaporation from the ocean surface. Sea-ice may also be a key factor, in e.g. leading to the (seasonal) rejection of dense salty brine (and removal of freshwater to the forming ice).

In turn, this suggests two sets of 'knobs' in the model that influence the patterns and magnitude of temperature and salinity across the ocean surface:

1. Continental Configuration.

The configuration of the continents and ocean basins will dictate the shape and location of the highest latitude ocean regions – presumably the locations where on average, deep-water formation will occur (and hence forming the downwards/sinking limb of the MOC).

The relevant model 'knob' is hence how the position and orientation of continental blocks is configured on the surface of the Earth. This (creating or changing the continental configuration used by **muffin**) can all done using the **MATLAB muffingen** program.

2. Global Climate.

The mean and climate as well as the zonal temperature gradients, together with whether or not sea-ice forms (and how much), will modulate both temperature and salinity patterns at the ocean surface.

The model knobs here are primarily:

- (a) Atmospheric pCO_2 , or in the absence of an explicit carbon cycle, a prescribed radiative forcing:

`ea_radfor_scl_co2=1.0`

²Note that if the *base-config* is the same as the previous experiment, but you have changed a parameter value (in the *user-config*), you do not need to re-compile.

which here, specifies $\times 1 CO_2$ equivalent radiative forcing.

- (b) One could also adjust the value of the solar constant (here, given with its modern/present-day default value):

```
ma_genie_solar_constant=1368.0
```

which has a subtly different effect from changing CO_2 radiative forcing, as radiative forcing has a relatively spatially uniform impact, whereas changing the solar constant has a disproportionate impact towards the Equator (where the incident solar shortwave radiation is the greatest). So changing the solar constant is likely to impact the pole-to-Equator temperature gradient. e.g. see: Lunt, D. J., A. Ridgwell, P. J. Valdes, and A. Seale, Sunshade World.: a fully coupled GCM evaluation of the climatic impacts of geoengineering, *GRL* 35, L12710, doi:10.1029/2008GL033674 (2008).

- (c) One could also adjust the planetary albedo, particularly in respect of looking to modify the pole-to-Equator temperature. In the **muffingen** created configurations, there is an explicit 1D file specifying the zonal (with latitude) planetary albedo profile:

```
xxxxxxxx.albd.dat
```

where `xxxxxxxx` is the 8-character name of the **muffingen** created World. It would be a simple matter of editing the values in the file (between the range of 0.0 and 1.0, for perfectly absorbing, and perfectly reflective, respectively).³ Note that the file data order is North-to-South (going from top to bottom in the file).

5.2.3 Gateways and barotropic flow

The overarching question in the context of ocean gateways and barotropic flow, is a little harder to define succinctly; it has to do with the details of the degree of alignment (or not) of the prevailing wind (stress) with gateways, and the degree to which this give rise to strong zonal ocean flow. A good and obvious modern example is the existence of the Drake Passage, between the northern tip of the Antarctic Peninsular, and the southern tip of Southern America, how this aligns with the prevailing Westerlies in the Southern Hemisphere, and hence the nature and strength of the ACC (Antarctic Circumpolar Current).

So the question naturally arises: how misaligned does a gateway have to be relative to the prevailing wind stress maximum, before the circumpolar (or circum-equatorial) flow ceases. What about having multiple gateways and their relative alignment, including what happens if one gateway is aligned with Westerly wind stress, and a second with Easterly wind stress – who ‘wins’)? Also – what about sill depths? For the Drake Passage, the ocean floor, while tectonically messy, is generally relatively deep. What happens to ocean circulation with a progressively shallow sill depth?

The 3 key controls in this exercise are then:

1. Gateway alignment.

The degree of alignment (or latitudinal correspondence) of a gateway with the maximum of the zonal wind stress. (For multiple gateways, what if one aligns with a wind-stress maximum of the opposite sign?)

2. Gateway width.

The width of a gateway (and how important this is in exerting control on ocean circulation).

³Best to make a copy of the original file before you modify it.

3. Sill depth.

The sill depth! This could be uniform in depth across the gateway (easiest/best) or could be varying (probably not so easy to learn anything).

The importance of the position of the gateways is primarily only in the context of the position of the maxima in the wind-stress field. In theory, the shape of the assumed zonal wind-stress could be altered ... but the wind stress needs to be defined on 2 different ocean grids ... and this gets messy ...

As before, there are a number of changes that can be made in the model to explore the consequences of varying gateway position and sill depth (and knobs, turned):

1. Land-sea mask.

The model knob is hence how the width and location of gateways has been defined in the land-sea mask. There is also the question of how many gateways (and their respective position).

As before, this (creating or changing the continental configuration used by **muffin**) is all done using the **MATLAB muffingen** program. Any of the given example **muffingen** configurations: drakeworld, eqpasworld, ridgeworld, or waterworld, could be taken as starting points – copying and renaming the configuration .m file, as well as the .dat file in the INPUT directory. Or start from scratch and a the 'blank' (initially all ocean) example configuration.

2. Ocean bathymetry.

The bathymetry (ocean floor depth) can be changed to alter the sill depth. The easiest way to do this is probably within the **muffingen** editor, ensuring the following input parameter setting is set:

```
opt_user=true; % [false/true] enable user input to grid
```

in the configuration .m file. (One might also then 'draw' in the gateways by hand at the same time.)

Note that a gap between 2 land masses must be 2 or more cells wide, to count as a gateway and allow barotropic flow in **muffingen**).

3. Wind stress strength.

Although it is messy to attempt to edit the profile of the applied wind-stress field, it is possible to scale its impact on ocean circulation lower and higher. The **muffin** parameter for this is:

```
go_13=1.531013488769531300
ea_11=1.531013488769531300
```

Somewhat bizarrely ... it appears twice ... with different parameter names. Both parameters must be changed to the same value. Place these lines (of the new parameter value assignments) in the *user-config* file for the experiment. Higher values result in a stronger applied wind-stress on the ocean surface.

Note that although this is the simplest change to make, it may not have such a clear scientific question associated with it (other than the obvious and trivial).

5.2.4 Analysis

How do you 'judge' the strength and characteristics of deep-water formation, global overturning, and circulation patterns and strength, in general, or their climate (or biogeochemical) impacts?

This is not a trivial question. Often, additional ocean tracers are employed in models to generate a quantitative measure of the age of a parcel of water (mean time since it last saw the surface), of some measure of the efficiency of ventilation, or large-scale transport at the surface or at depth.⁴ Or rather involved analysis of ocean physics and transport might be employed.

One starting point is to read some of the literature where the climate properties of various hypothetical worlds have been investigated. Such as:

- Marshall *et al.* [2007] – ‘Mean Climate and Variability of the Atmosphere and Ocean on an Aquaplanet’, *Journal of the Atmospheric Sciences* **64**.
- Enderton and Marshall [2009] – ‘Explorations of Atmosphere–Ocean–Ice Climates on an Aquaplanet and Their Meridional Energy Transports’, *Journal of the Atmospheric Sciences*.
- Ferreira *et al.* [2010] – ‘Localization of Deep Water Formation: Role of Atmospheric Moisture Transport and Geometrical Constraints on Ocean Circulation’, *Journal of Climate* **23**.
- Smith *et al.* [2006] – ‘Global Climate and Ocean Circulation on an Aquaplanet Ocean–Atmosphere General Circulation Model’, *Journal of Climate* **19**.

In addition, the sub-sub-sections that follow outline some simple diagnostics and ways of going about some quantitative analysis.

Simple/global diagnostics

- In the biogem output results folder, there is a time-series file named:

`biogem_series_misc_opsi.res`

This contains a summary of the evolution with time, of the minimum and maximum (anywhere) global overturning stream-function values.

- Another simple property of the climate system that you might consider, is the pole-to-equator temperature gradient, both in terms of atmospheric temperature, and ocean surface temperature⁵ (although they should presumably be closely coupled).

Why? Because the large scale (overturning) circulation of the ocean should be transporting heat from the high latitude surface to the deep ocean. This presumably would act to reduce latitudinal temperature gradients. In contrast, a strong zonal flow might prevent latitudinal transport of heat.

Using Panoply

In a physics-only ($T + S$ tracer) configuration of **muffin** you are somewhat limited in what you can look at. A good starting point is the previous tutorial on ocean circulation and AMOC stability (in the context of the modern world, climate and continental configuration). For instance, you know already how to plot and visualize the MOC using **Panoply**. The Atlantic basin and hence the existence of an AMOC, is pretty specific and unique to the modern world, so likely you’ll need to focus on the global MOC. (A good starting point is to familiarise yourself with the pattern and intensity of the modern.)

For example – the drakeworld configuration gives rise to a global MOC pattern as shown in Figure 5.1.

This is stored as the variable `phys_opsi` in the netCDF file `fields_biogem_2d.nc`. Remember that the first time-slice plotted in **Panoply** is the first time-slice saved and it may also be up-site-down(!) by default :(.

⁴We’ll see such tracers employed later in the course.

⁵**Panoply** has an option for plotting the zonal mean, from which you could read off pole-to-equator temperature gradients.

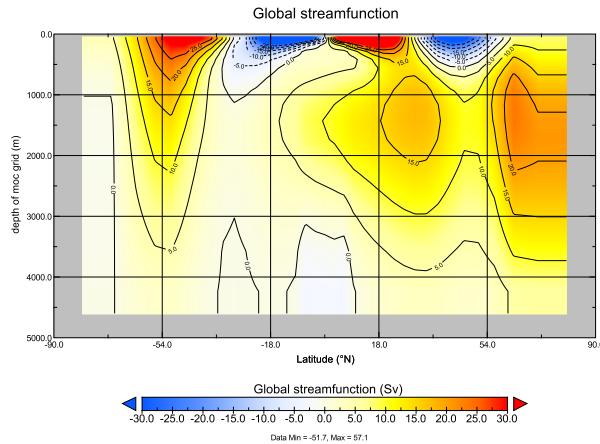


Figure 5.1: Drakework world overturning circulation.

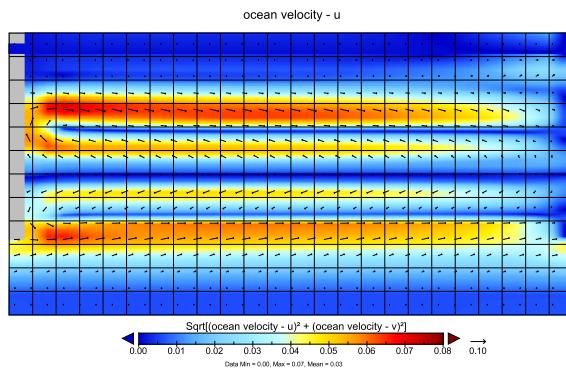


Figure 5.2: Surface current velocity field (arrows) plus speed (color scale).

You can also plot the barotropic circulation using **Panopoly** – the variable is called `phys_psi` – this is shown in Figure 5.3.

Finally in **Panopoly**, you might plot the current field (surface or otherwise), as per Figure 5.2.

Using MATLAB ...

TO COME ...

Via color tracing

Numerical color tracers can be used (as you have seen previously) to trace flow-paths. The default `base-config` files provided as part of the **muffingen** software, define a configuration with only 2 tracers in the ocean – T and S. To add red and blue dye tracers, in the `base-config` file, find the section marked:

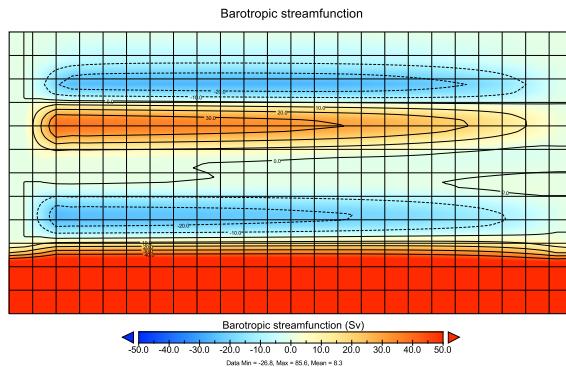


Figure 5.3: Drakeworl world barotropic circulation.

```

# ****
# TRACER CONFIGURATION
# ****
# the total number of tracers includes T and S
# T and S do not need to be explicitly selected and initialized
# ****
# Set number of tracers
GOLDSTEINNTRACSOPTS='$(DEFINE)GOLDSTEINNTRACS=2'
# list selected biogeochemical tracers
# <<<                                >>>
# list biogeochemical tracer initial values
# <<<                                >>>
# ****

```

You need to make several (but simple) edits here:

1. Firstly, change the number of tracers from 2 to 4, in line:

```
GOLDSTEINNTRACSOPTS='$(DEFINE)GOLDSTEINNTRACS=2'
```

i.e. to make:

```
GOLDSTEINNTRACSOPTS='$(DEFINE)GOLDSTEINNTRACS=4'
```

2. Secondly, you need to list the additional list selected biogeochemical tracers.

For this add the following code in place of the (<<< >>>) line:

```
gm_ocn_select_48=.true.
gm_ocn_select_49=.true.
```

3. Lastly, under the section list biogeochemical tracer initial values, and in place of the (<<< >>>) line:

```
bg_ocn_init_48=0.0
bg_ocn_init_49=0.0
```

To apply a 'red' dye tracer to the ocean, you can employ the same *forcing* as used in the experiments in Chapter 3 – *pyyyyz.Fred* and implement this with similar user-config settings:

```

#
# --- FORCINGS -----
#
bg_par_forcing_name="pyyyyz.Fred"
```

```
bg_par_force_point_i=1
bg_par_force_point_j=1
bg_par_force_point_k=1
bg_par_ocn_force_scale_val_48=0.0
```

However, the specific i,j,k (longitude, latitude, ocean depth layer) model grid locations will depend on the resolution of the World you generated or had adopted, and it may have e.g. 18 or 36 maximum i,j values, and generally, either 8 or 16 maximum depth levels (k). (The values shown would be for the southern and western corner of the grid at the bottom of the ocean ... but note that this location might be land or seafloor in your particular World!)

Via age tracing

Refer to the 'HOW-TO' in the muffin manual – 'Add a water mass age tracer':

"Water mass age tracing can be configured quite simply, and with the tracer concentration manipulations carried out automatically, as follows. As base-config file with both red and blue color tracers defined is still required, but rather than have to define and use a set of forcings (and associated forcing configuration), a single parameter is added in the user-config file:

```
bg_ctrl_force_ocn_age=.true.
```

This will automatically create the age tracer and additional explicitly output (in netCDF) both the total age of a water parcel, as well as the age relative to the surface (ventilation age). In this methodology, in the netCDF output, the concentration ratio of blue/red, should be 'age' – the mean time that a parcel of water was last at the surface."

Note that an experiment spin-up duration of perhaps as much as 10,000 years is required in order to obtain a stable tracing of water mass age.

Exploring the consequences of fossil fuel

CO_2 emissions

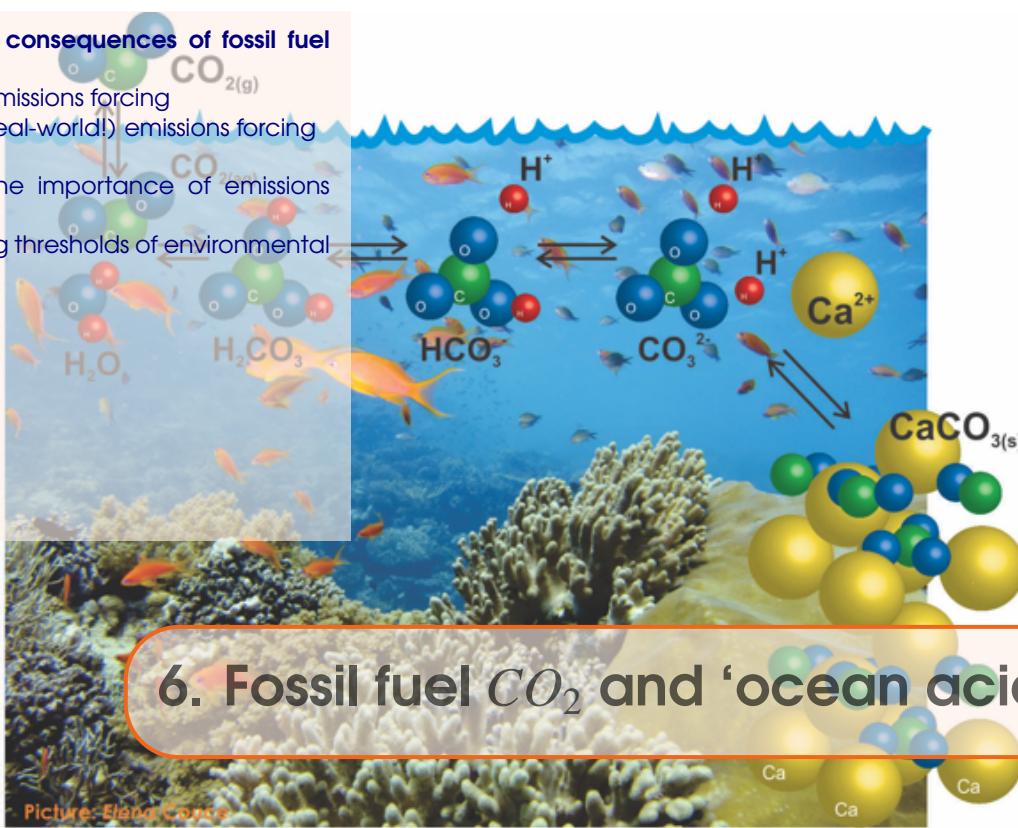
Idealized emissions forcing

Historical (real-world) emissions forcing

Further ideas

Assessing the importance of emissions rate

Determining thresholds of environmental impact



6. Fossil fuel CO_2 and 'ocean acidification'

Picture: Elena Correa

Readme

You will need to download a new *restart* file prior to embarking on the experiments. This pre-industrial spin-up includes a basic ocean (-atmosphere) carbon cycle plus various diagnostic anthropogenic tracers, following *Cao et al.* [2009].

To fetch this: change to the cgenie_output directory, and type:

```
$ wget http://www.seao2.info/cgenie_output/LAB_3.SPIN.tar.gz
```

Extract the contents of this archive by typing:

```
$ tar xfzv LAB_3.SPIN.tar.gz
```

(change directory back to genie-main to run the model)

6.1 Exploring the consequences of fossil fuel CO_2 emissions

For the next experiment(s) you can chuck CO_2 into the atmosphere, just for the hell of it. As much as you want! Apparently, humans are actually doing this now. Imagine that!

A new *user-config* for **muffin** – LAB_3.CO2emissions – is provided and configured with climate being responsive to CO_2 (i.e., it takes account of CO_2 -climate feedbacks):

```
# set CO2-climate feedback
ea_36=y
```

Note that although a biological scheme is set up, with the fixation of organic carbon at the surface and sinking into the ocean interior, plus a rate of calcification by plankton at the surface ocean that is responsive to ocean acidification and saturation state (i.e., it takes into account CO_2 -calcification feedbacks, which will additionally interact with climate – see *Ridgwell et al.* [2007b, 2009]¹) ... we will concentrate only on the carbon geochemistry. A 'biological pump' in the ocean is set up simply to create a modern-like ocean surface carbonate chemistry. How the biological pump actually operates, response to changes in e.g. climate and nutrient availability, we'll consider in another chapter.

In this *user-config* file, a release of CO_2 to the atmosphere is prescribed, which by default is set to a value of just 1 PgC and over an interval of just a single year. (Releasing CO_2 just over a single year is obviously rather unrealistic and many impacts will decay away rapidly, but represents a useful idealized experiment for assessing the time-scale(s) of fossil fuel CO_2 uptake by the ocean.) Additional (netCDF) output has also been prescribed, via the setting: `bg_par_data_save_level=10` (see Section 12.4) so that more information relevant to assessing ocean acidification is saved.

Run the experiment for e.g., 20 (or more if you like) years, starting from the pre-industrial re-start experiment LAB_3.SPIN, i.e.:

```
$ ./runmuffin.sh cgenie.eb_go_gs_ac_bg.worjh2.BASE LABS
LAB_3.CO2emissions 20 LAB_3.SPIN
```

As for what model results variables to consider ... think about the climate change and ocean acidification literature and which environmental (physical and geochemical) properties are considered either critical for ecosystems or are simply helpful and illustrative. Refer to Section 12.6.2 for a summary of some of the key ocean acidification (and other) variables that may be saved by the model². In the 3-D netCDF time-slice file remember, for instance, that ocean surface waters in which aragonite becomes under-saturated ($OHMEGA < 1.0$) is regarded as a critical threshold for organisms making aragonite shells and skeletons and spells TROUBLE for some poor calcifying marine organism somewhere. (Temperature is also highly relevant to marine ecosystems under future global change.) Note that the calcification response is encoded in the model and described in *Ridgwell et al.* [2007a,b] and may or may not reflect the Real World.

For climate change ... the variables of particular interest should be obvious. Remember that there are both *time-series* outputs, as well as 2D and 3D fields, any or all of which might be helpful for elucidating impacts.

¹e.g. obtain from <http://www.seao2.info/pubs.html>

²(depending on specific data saving configuration)

6.1.1 Idealized emissions forcing

You can easily modify the experimental design to release more/less CO_2 very much as you did for the red dye tracer. In the *user-config* file, the line:

```
bg_par_atm_force_scale_val_3=8.3333e+013
```

scales the time-history of the CO_2 flux, given in the forcing file:

```
biogem_force_flux_atm_pCO2_sig.dat
```

... which can be found in the directory:

```
cgenie.muffin/genie_forcings/pyyyz.FpCO2_Fp13CO2
```

The format of this file is:

```
-START-OF-DATA-
```

```
0.0 1.0
1.0 1.0
1.0 0.0
999999.9 0.0
```

```
-END-OF-DATA-
```

and defines an emission of $1molC$ (carbon) per year over the first year (year 1.0) of the model experiment (between year 0.0 and 1.0), but which in the example *user-config* is then scaled by a value of 8.333×10^{13} (by the parameter *bg_par_atm_force_scale_val_3*) to give a total of $1PgCyr^{-1}$. (Year 999999.9 has no special meaning and is simply just way in the future ...)

Pause ... and note briefly how the final CO_2 flux is arrived at. **muffin** calculates it by multiplying the value in the forcing file (1.0) by a modifying parameter in the *user-config* file ($8.3333e+13$). The total flux is hence: $1.0 \times 8.333 \times 10^{13} = 8.333 \times 10^{13} molCO_2 yr^{-1}$. If you set both values as 1.0, you’d get very little carbon released (a single mol!). If you screw up and multiply $8.3333e+013$ and $8.3333e+013$ together as the total flux ... you’ll soon know it as you cook the Earth ... But it does not matter which parameter has value 1.0 and which scales the units ($8.3333e+013$). For now, it is simply more convenient to be able to edit the *forcing* file with ‘simple’ numbers (and leave the large numbers and units conversion in the *user-config* file).

Together, the scaling and forcing value gives a CO_2 release of $1PgCyr^{-1}$ for just a single year compared to current emissions are about $10PgCyr^{-1}$. So, do not expect anything exciting to happen at this point.

(The parameter: *bg_par_atm_force_scale_val_4=-27.0* specifies the carbon isotopic composition of fossil fuel carbon and can be ignored for now.)

Because ‘accidents can happen’ and the global environmental changes induced by the massive fossil fuel CO_2 release can obscure mistakes made in the experiment configuration (parameter values) and/or the re-start used, you are strongly advised to first (or in parallel, as a job submitted to the cluster – refer to Lesson Zero to remind yourself of the command line syntax needed for this) – and run a control experiment:

```
$ ./runmuffin.sh cgenie.eb_go_gs_ac_bg.worjh2.BASE LABS
LAB_3.CONTROL 20 LAB_3.SPIN
```

Here – the *user-config* defining the control experiment (LAB_3.CONTROL) is identical to that for the actual experiment itself (LAB_3.CO2emissions) with the exception of the scaling of the CO_2

emissions that is set to zero. (It is left completely to you to create the experiment configuration file `LAB_3.CONTROL`.)

If everything is OK, atmospheric CO_2 (and climate) following from the *re-start* should be stable and there should be little (or no) drift in any of the output variables (because the *spin-up* you are re-starting from should have been run to an equilibrium state and you have not changed anything in the control experiment, right?).

It is good practice (i.e., do it!) to always run a control experiment for each different type of experiments – e.g., ideally run one control experiment for each set of CO_2 emissions experiments.

When you have run both the real and control experiment, compare the results. View (or plot) both relevant *time-series* output, and create anomaly maps of key *time-slice* variables in **Panoply** or **MATLAB**, using a corresponding *time-slice* from the control experiment to create the experiment anomaly with.

OK. You might want to run something a little more exciting now. For instance, rather than

```
-START-OF-DATA-
 0.0  1.0
 1.0  1.0
 1.0  0.0
999999.9  0.0
-END-OF-DATA-
```

you might have:

```
-START-OF-DATA-
 0.0  1000.0
 1.0  1000.0
 1.0      0.0
999999.9      0.0
-END-OF-DATA-
```

So now a total of 1000 PgC over a single year. Now you should see some policy-relevant impacts occur :o)

(Again, contrast control and experiment results to quantify/visualize the impacts of the CO_2 release.)

You can control the shape of the emissions profile as well as its magnitude. Between the start and end ‘tags’ in the text *forcing* file, the data is arranged into 2 columns: the first contains a series of tie-points for defining the timing of changes in emissions, and the 2nd column contains flux information (units of $PgCyr^{-1}$ when scaled by the parameter `bg_par_atm_force_scale_val_3` in the *user-config*). At each time-step of the model, the CO_2 flux to be applied to the atmosphere is interpolated between these time points.

For instance, in the *forcing* (directory) file biogem_force_flux_atm_pCO2_sig.dat, the purpose of:

```
0.0 1.0
1.0 1.0
1.0 0.0
999999.9 0.0
```

is to specify a uniform flux of 1.0 (scaled to $PgCyr^{-1}$) over the first full year of the model run, followed by a sharp turn-off to zero flux at the end of first year (and remaining zero thereafter). To extend the period of emissions – for example:

```
0.0 1.0
10.0 1.0
10.0 0.0
999999.9 0.0
```

would result in a uniform flux lasting 10 years with a sudden cut-off and zero thereafter (i.e., once scaled by the parameter in the *user-config* – $1PgCyr^{-1}$ over 10 years – $10PgC$ total emissions). In contrast;

```
0.0 0.0
10.0 1.0
10.0 0.0
999999.9 0.0
```

would result in a linear ramp, starting from zero at the start of year 0.0 to $1.0PgCyr^{-1}$ at year 10.0 and then suddenly ceasing and remaining at zero for the remainder of the experiment (a total CO_2 emission of $1 \times 1.0 \times 0.5 = 5PgC$ over 10 years).

Make up a few ‘shapes’ (and hence different experiments), maybe for the same integrated/total emissions, explore the effect of different rates of rise/fall in the release rate. And/or for the same release rate and/or duration, explore the impact of different total emissions of CO_2 . Try and think in terms of hypotheses and formulate questions to guide your experimental design/configuration. (An alternative approach is to create random scenarios and in the analysis, fish for interesting patterns that could lead to knowledge and/or specific questions to be tested further, but it is better to start off with a hypothesis in mind.)

Note that you can either edit and re-use the same *forcing* directory and name, modifying the file biogem_force_flux_atm_pCO2_sig.dat each time but then losing an explicit record of how you might have set the emissions profile previously), or you can copy and rename the entire *forcing* directory (and then edit biogem_force_flux_atm_pCO2_sig.dat). In the *user-config*, you then need to specify this new forcing (directory) name, e.g.:

```
# specify forcings
bg_par_forcing_name="pyyyyz.FpCO2_Fp13CO2.NEW"
```

if you, for instance, called your new *forcing* directory (in genie-forcings): pyyyz.FpCO2_Fp13CO2.NEW.³

³Refer to the directory map in Figure 1.1. if in doubt here.

6.1.2 Historical (real-world!) emissions forcing

Historical and future (e.g. IPCC 'SRES') emissions scenarios can also be prescribed explicitly. A historical emissions forcing (technically: a prescribed concentration profile of pCO_2 and other anthropogenic gases) can be specified by adding/substituting the following lines to the *user-config*:

```
bg_par_forcing_name='worjh2.historical2010'
```

Now, no additional scaling is needed because the forcing specification directly follows the observed change in atmospheric concentration with time (in units of atm CO_2), and the line containing the scaling parameter:

```
bg_par_atm_force_val_time_3=1.0
```

should be deleted (or commented out with a # at the start of the line, as the default value is 1.0). For completeness, you might also set the scaling of atmospheric CO_2 $\delta^{13}C$ ⁴ to a value of 1.0, even if we are not interested in $\delta^{13}C$ at this time ...

An additional line is now needed in the *user-config* because the historical pCO_2 transient starts in the 1700s (for which a nominal date of 1765 is often used) rather than year zero. For example, to start **muffin** from year 1765 rather than zero, the start year parameter must be set:

```
bg_par_misc_t_start=1765.0
```

Now because the start year has changed ... it is convenient to specify a set of time save points that are consistent with the historical period, e.g. you might add the parameter settings:

```
bg_par_infile_slice_name='save_timeslice_historicalfuture.dat'
bg_par_infile_sig_name='save_timeseries_historicalfuture.dat'
```

which specify a series of time points at which data is saved that aligns with historically relevant years. View the contents of these (text) files, which can be found in the directory *genie-biogem/data/input*, to get a sense of how frequently and when data will be saved, and how this differs from the default settings, which are defined in the files:

```
save_timeseries.dat
save_timeslice.dat
```

A *user-config* with these specific changes is provided for your convenience (or to double-check you were following it all) -- *LAB_3.historical*. A suitable experiment would then be one run for 245 years so that it reaches year 2010 (having started from year 1765):

```
$ ./runmuffin.sh cgenie.eb_go_gs_ac_bg.worjh2.BASE LABS
    LAB_3.historical 245 LAB_3.SPIN
```

(Here, the 245 year run duration is simply calculated to get you from year 1765 to 2010 ...)

WARNING! Ignore the 'WARNING's at the start – these are simply telling you that more tracer forcings have been specified than you have selected tracers for in the *base-config* (*cgenie.eb_go_gs_ac_bg.worjh2.BASE*). (A different *base-config* with additional selected tracers could have been specified to make use of other historical changes in atmospheric composition, such as of radiocarbon (^{14}C) and CFCs.) Also: from year 1765 onwards, changes in atmospheric CO_2 only rise very slowly initially. Don't expect to see anything happen in 10 seconds flat because relatively few people and countries in the 1800s could be bothered to burn much more than

⁴bg_par_atm_force_scale_val_4

a little local coal. You could potentially start your experiment at year 1850, changing the value of `bg_par_misc_t_start` and specifying shorter experiment duration if you are desperate for the End of the World to come.

Don’t forget: you could submit this experiment to the cluster and do more (idealized emissions) ‘playing’.

Given that there is observationally-based information on the distribution of anthropogenic CO_2 taken up by the ocean (e.g., *Sabine et al.* [2004]) ... and you are running a historical transient experiment with the model driven by observed increases in atmospheric pCO_2 ... you are in a position to critically evaluate the models ability (or lack of) to represent the future-critical process of oceanic fossil fuel CO_2 uptake and transport by large scale ocean circulation.

In the 2D netCDF output, there is a variable for the water column integrated inventory of DIC – equivalent to the Sabine map except you will need to subtract the preindustrial background of DIC first, i.e., to create a DIC anomaly map representing only the added fossil fuel CO_2 component of ocean DIC. The data in the Sabine paper clusters around 1994. A *time-slice* centered on this year (1994.5) has been configured in the model exactly for this purpose. Your baseline state can either be from prior to CO_2 emissions commencing at any significant rate (e.g., 1750.5) or (better), from a control experiment. Note that similar comparisons could be (and are regularly) made with other tracers such as CFCs, which provide additional insights into the patterns and time-scales of trace gas update and ocean circulation. (See: *Cao et al.* [2009])

Observational data, re-gridded to the **muffin** grid and in netCDF format can be downloaded from the ‘usual place’ (<http://www.seao2.org/mymuffin.html>) from the ‘got data?’ box on the left. You could for instance, compare horizontal or vertical slices (3D netCDF) and create difference (anomaly) maps. Somewhat more representative of the entire ocean is to compare (or calculate difference maps) of zonal average profiles. Unfortunately, the observations are not in the form of water column integrals and hence you cannot create difference maps of model as per the *Sabine* paper ... unless you are **MATLAB**-friendly and you use the 3D **BIOGEM MATLAB** plotting scripts (<https://github.com/derpycode/muffinplot>) whose use is somewhat described in the **muffin** user-manual. Examples of **MATLAB** plotting of the model vs. observed anthropogenic anomaly are shown in Figure 6.1 (note the use of plotting un-interpolated model grid data as colors but with an interpolated contour overlain to help guide the eye and pick out features).

Finally, and the closest to being slightly interesting: rather than applying highly idealized pulses of CO_2 emissions, the IPCC ‘SRES’ emissions scenarios can be used to make future projections with. An example forcing of this sort is provided and can be selected by changing the name of the forcing selection parameter (`bg_par_forcing_name`) to:

```
worjh2.FeMahowald2006.FpCO2_Fp13C02_A2_02180PgC
```

which gives you the IPCC ‘A2’ scenario (extended beyond year 2010 in this case to give a total cumulative fossil fuel burn of 2018 PgC, e.g. *Ridgwell and Schmidt* [2010]). Again, as this forcing has units of $PgCyr^{-1}$ in its time-series file, you will need to add a scaling parameter to the *user-config* file to turn units of $PgCyr^{-1}$ into $molCyr^{-1}$, i.e.,

```
bg_par_atm_force_scale_val_3=8.3333e+013
```

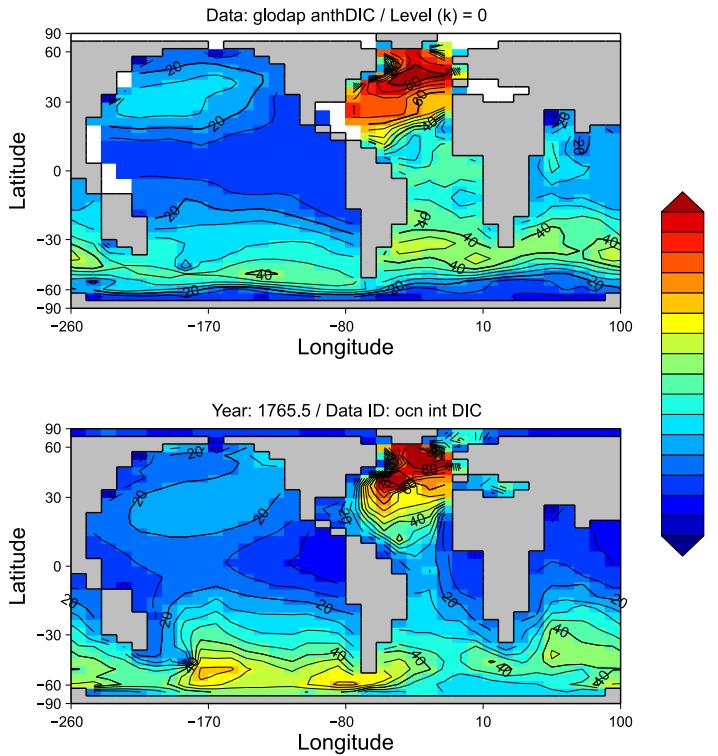


Figure 6.1: Observed (top) vs. Model (bottom) anthropogenic CO_2 inventories. Data and model water column integrals in units of mol CO_2 m^{-2} and are nominally with respect to year 1994.

For complete ‘realism’, you will need to run this experiment starting from the end of the historical transient experiment you have just run (Section 1.6), e.g.,

```
$ ./runmuffin.sh cgenie.eb_go_gs_ac_bg.worjh2.BASE LABS
LAB_3.future 90 LAB_3.historical
```

and with the start from year now set to year 2010 (the end year of the historical transient):

```
bg_par_misc_t_start=2010.0
```

Note that the *user-config* `LAB_3.future` is not provided for you – you will need to create this (or a file named whatever you like) by copying e.g., `LAB_3.EXAMPLE` and making the parameter changes described above (forcing specification parameter, emissions scaling parameter, and start year parameter).

You can also easily replace the details of the emissions with other SRES scenarios – simply find the year *vs.* emissions rate information from the interweb⁵ and edit or copy-and-paste the flux values for each decade into the file `biogem_force_flux_atm_pCO2_sig.dat` in the forcing directory. **muffin** will then automatically interpolate between the decadal tie-points to give a continuous change in emissions. Now you are able to make a rather more realistic/plausible assessment of when and where potential ecological impacts (via assumed ocean chemistry criteria) might occur.

Try running (e.g. as jobs submitted to the cluster queue), some other actual or made up SRES emissions scenarios.

⁵e.g., http://sres.ciesin.columbia.edu/final_data.html

6.2 Further ideas

6.2.1 Assessing the importance of emissions rate

By editing the flux and/or timing information you can control the CO_2 emissions trajectory and total fossil fuel burned. Explore different CO_2 release assumptions and note their impact on climate and ocean biogeochemistry. Much more realistic and appropriate to our *current* global experiment is a lower rate (order of 10 or 20 $PgCyr^{-1}$) released over a longer interval (order of 100 years) compared to the conceptual 1000 PgC near-instantaneous pulse. Because the experiments are getting longer to run in real time ... remember to make appropriate use of the cluster queuing facility – i.e., think about whether you want to sit around starting at the screen for 15 minutes waiting for a new line of numbers appear – if not: submit to the cluster queue. For instance, one might try and address the question: “For a given total release of fossil fuel CO_2 , is it safer to burn it slower?” The answer is maybe not completely obvious, as burning carbon resources slower will result in a small global impact, but perhaps one that persists for longer. You could conceive of an ensemble (set) of model experiments, maybe one of $100 PgCyr^{-1}$ for 1 yr, one of $10 PgCyr^{-1}$ for 10 years, and one of $1 PgCyr^{-1}$ for 100 years, and run them all for e.g., 100 years.⁶ (As jobs submitted to the queue, all can be run simultaneously (with blank spaces between commands!).) (Don’t forget the control experiment!)

Note note that you should create 3 new *forcings* based on the original if you are editing the same original *forcing* and expecting to run different ones at the same time. Really, this is little more than copying and renaming *user-config* files, except it involves entire directories in *genie-forcings*. Remember that the forcing is specified by the directory name assigned to *bg_par_forcing_name* (enclosed in ‘’).

6.2.2 Determining thresholds of environmental impact

There are various concerns about the impacts of continuing fossil fuel CO_2 emissions and a number of proposed climatic (e.g., the 2 degree C global warming limit often mentioned in policy documents) and ecological ‘tipping points’. You can assess the maximum allowable CO_2 emissions to remain within particular global environmental limits in the model. For example:

- What is the maximum total CO_2 release that can be made without inducing aragonite undersaturation at the ocean surface anywhere (or any season – see Section 5.2.3 in the User Manual for seasonal time-slice data saving)? How important is the time-scale of emissions in determining this? For total emissions above this: where in the ocean does the surface first become under-saturated? How large would the emissions have to be in order to induce undersaturation at the surface in the tropics (home to socio-economically important reef systems). These are questions that can be addressed with simple CO_2 release experiments in ocean carbon cycle models and everyone seems to get a GRL paper out of it each and every time!
- How important are CO_2 -climate feedbacks in amplifying or diminishing future climate and ocean carbonate chemistry changes – e.g., is the same atmospheric pCO_2 value reached with and without climate feedback (and surface warming) – if not, why? You can investigate this by contrasting an experiment made including CO_2 -climate feedback with one made without. The CO_2 -climate feedback can be turned off by setting: `ea_36=n`.

⁶These all represent rather unrealistically small total CO_2 releases and you may want to consider a total more like 1000 PgC or rather more. You may also want to think about more realistic shapes rather than pulses, such as some sort of ramp up and then down in the emissions rate (but for the same total emissions).

- Also: How much CO_2 emission does it take to significantly ‘collapse’ the AMOC and over what time-scale? (Or alternatively: what is the atmospheric pCO_2 threshold for collapse?) If the AMOC weakens or collapses ... why in the absence of a prescribed freshwater perturbation does this happen? What physical process are at play in response to rapid CO_2 release too the atmosphere that may act to reduce or shutdown deep-water formation in the ocean model? (Plotting appropriate ocean property anomalies between the CO_2 release experiment and a control experiment might help.)

Experiments could be hypothetical and consisting of CO_2 pulses or ramps (or exponentials) and run on directly from a pre-industrial spin-up, or more ‘realistic’ and run on from the end of a historical transient experiment (e.g., starting in year 2010).

xxx

Ocean carbon geoengineering

- Iron fertilization
- Phosphate fertilization
- Enhanced weathering
- Modifying spatial patterns

Further ideas

- Impacts to look out for
- Other thoughts and suggestions
- Further modifications of the biological pump in the ocean

7. Ocean biogeochemical cycles

READ.ME

You will need to download a new *restart* file prior to embarking on the experiments. This differs from previous provided restarts in that it now includes an iron cycle in the ocean and hence co-limitation of biological productivity by Fe.

```
$ wget http://www.seao2.info/cgenie_output/LAB_4.SPIN.tar.gz
```

Extract the results in the usual way and in the usual place ... and return to *genie-main*.

7.2 Ocean carbon geoengineering

In this Section you'll use the 'excuse' of geoengineering, to perturb and elucidate the response of the global carbon cycle and climate system and hence learn something new/different about Earth system dynamics compared to e.g. its response only under standard future carbon emissions scenarios.

In the following experiments you are going to explore some of the ocean biological controls on atmospheric pCO_2 (plus ocean acidification, and the distributions and intensities of oxygen minimum zones). Really, the 'geoengineering' focus is just an excuse to be looking at how the biological pump in the ocean works, how it regulates atmospheric pCO_2 , how sensitive it is to perturbation and what the consequences are of any changes in it. So if you are uncomfortable with ideas of large scale manipulating the Earth system, you might instead think about the relevance of the experiments to e.g. understanding why atmospheric pCO_2 was low at the time of the last glacial.

The overall idea of this Chapter is to run future CO_2 emissions scenarios and test whether ocean carbon geoengineering is an effective means for reducing future ocean acidification and marine ecological impacts (but keeping in mind that you are also exploring the basic natural operation of the system in doing so). You will require a pre-industrial *spin-up* and will need to create a new historical pCO_2 transient experiment because you are now using a different *base-config* (`cgenie.eb_go_gs_ac_bg.worjh2.BASEFe`) that includes additional tracers for the marine iron cycle, i.e. you cannot simply use any of the experiments from previous labs as a restart.

Refer to Section 12.6 for a guide as to what to 'look for' in the model results.

So to start with, go ahead and run a new historical transient experiment. A *user-config* is provided for your convenience (`LAB_4.historical`) ... but maybe check the settings for e.g. start year, as well as note that there are a number of new parameters to control the iron cycle (amongst other differences) as compared to before:

```
$ ./runmuffin.sh cgenie.eb_go_gs_ac_bg.worjh2.BASEFe LABS
    LAB_4.historical 245 LAB_4.SPIN
```

(ignore the 'WARNING's at the start)

The provided example *user-config* – `LAB_4.EXAMPLE` – includes parameter settings for controlling any one of 3 different possible ocean carbon geoengineering schemes, described below. By default, these are commented out (== ignored by the model) and only the forcing for the A2 emissions scenario (`worjh2.FeMahowald2006.FpCO2_Fp13CO2_A2_02180PgC`) with no geoengineering enacted, is enabled by default. You might regard this as a control (reference) experiment for all the with-geoengineering experiments you might run, i.e. the impacts of CO_2 emissions in the absence of any mitigation by geoengineering. To activate any particular geoengineering forcing: simply un-comment (delete the # at the start of) the appropriate pair of lines (the first line being the forcing specification, and the second one the total flux forcing used in the geoengineering scheme). If you have multiple (un-commented) settings of a parameter (e.g. `bg_par_forcing_name`), then the value specified in the last occurrence of the parameter, is the one that is applied. This can get confusing, so if you un-comment out one set of parameter options, comment out (add a # to) the ones you are not using.

The geoengineering (and control) experiments need to be run starting from the end of your historical transient experiment:

```
$ ./runmuffin.sh cgenie.eb_go_gs_ac_bg.worjh2.BASEFe LABS
LAB_4.EXAMPLE 90 LAB_4.historical
```

Because with a modern configuration and additional tracers in the ocean, the model is running rather slower than in some earlier exercises, you may not want to run beyond the end of the century (hence the 90 year experiment duration, starting from year 2010, as suggested above).

Each of the example geoengineering scenarios are delineated by its own specific forcing – a set of files that live in a uniquely named sub-directory within genie-forcings. The three forcings are:

- worjh2.FeMahowald2006.FpCO2_Fp13CO2_A2_02180PgC_FFe
- worjh2.FeMahowald2006.FpCO2_Fp13CO2_A2_02180PgC_FP04
- worjh2.FeMahowald2006_FpCO2_Fp13CO2_A2_02180PgC_FALK

Each forcing includes the A2 CO_2 emissions scenario, with the annual emissions (CO_2 flux) `biogem_force_flux_atm_pCO2_sig.dat` in units of $PgCyr^{-1}$ (== GtC yr-1), hence requiring a units conversion setting in the *user-config* (`bg_par_atm_force_scale_val_3=8.3333e+013`) that is provided for you under the heading # CO₂ emissions scaling. (You can completely ignore the carbon isotope settings.)

Each forcing also includes a prescribed dust flux to the ocean surface (the FeMahowald2006 part of the directory name string). This is necessary because the model configuration you are using includes a co-limitation of biological productivity by iron (Fe) in addition to phosphate (PO_4). (The files associated with the dust forcing are: `biogem_force_flux_sed_det_sig.dat` and `biogem_force_flux_sed_det_SUR.dat` but you do not need to edit these files.) For the role of iron in controlling ocean productivity: possible starting points for background reading are: *Ridgwell and Kohfeld* [2007] (PDF available from my website) or *Jickells et al.* [2005] (Science).

The specific details of the 3 different example geoengineering scenarios are as follows:

7.2.1 Iron fertilization

Forcing: `worjh2.FeMahowald2006.FpCO2_Fp13CO2_A2_02180PgC_FFe`

– a constant (with time) flux of dissolved Fe (in addition to whatever Fe dissolves into the surface ocean from the dust flux) is specified in: `biogem_force_flux_ocn_Fe_sig.dat`. The magnitude of the applied flux is then scaled in the *user-config* file by the setting:

`bg_par_ocn_force_scale_val_9=1.0e+09`

Note that this is simply an example total global flux. You might consider higher or lower fluxes, as well as potentially how ‘practical’ the annual production and supply of such quantities might be.

A spatial pattern of the flux is also defined, in the file:

`biogem_force_flux_ocn_Fe_SUR.dat`

An example pattern is set (see later for instructions on modifying this) with a row of grid cells marked along the same latitude in the Southern Ocean. You do not need to retain this pattern. In choosing an alternative: think about where in the modern ocean biological productivity is thought to be at least partly limited by the availability of dissolved Fe. Remember that the model may or may not correspond with reality, i.e. it may or may not predict Fe limitation in the correct regions, which may affect your choice of location for iron fertilization.

7.2.2 Phosphate fertilization

Forcing: worjh2.FeMahowald2006.FpCO2_Fp13CO2_A2_02180PgC_FPO4 ('macro-nutrient' addition)

– a constant (with time) flux of dissolved PO_4 is specified in: `biogem_force_flux_ocn_P04_sig.dat`. The magnitude of the applied flux is then scaled in the *user-config* by the setting:

```
bg_par_ocn_force_scale_val_8=2.0e+12
```

Again, you should consider this as an example total flux. In choosing a total flux to apply, points of comparison include whatever the total weathering flux (via rivers) of P to the global ocean is. Also: global phosphate (fertilizer) production, which produces an interesting potential conflict between geoengineering and food production, although there are proposals for using fertilized ocean regions for enhanced fish production.

A spatial pattern of the flux is also defined, in the file:

```
biogem_force_flux_ocn_P04_SUR.dat
```

An example pattern has been set – here, the Equatorial Atlantic. In choosing your regions(s), think about where in the ocean (again – there may be differences between real ocean and model) productivity is currently limited by PO_4 . Also be aware of possible on-set of Fe limitation if you relieve the PO₄ limitation (i.e., you could potentially lose effectiveness if you supply too much PO₄ and instead productivity and CO_2 draw-down is capped by a second factor). You could potentially consider PO₄ and Fe addition at the same time ... ?

7.2.3 Enhanced weathering

Forcing: worjh2.FeMahowald2006.FpCO2_Fp13CO2_A2_02180PgC_FPO4 (alkalinity addition)

– a constant (with time) flux of alkalinity is specified in: `biogem_force_flux_ocn_ALK_sig.dat`. The magnitude of the applied flux is then scaled in the *user-config* by the setting:

```
bg_par_ocn_force_scale_val_12=5.0e+13
```

Again, another example total flux. In choosing a total flux to apply, points of comparison include whatever the total weathering flux (via rivers) of alkalinity (often described in terms of the bicarbonate ion flux) to the global ocean is. Also: global cement (lime) production. (Note that in one mole of lime: CaO , you have 2 moles of alkalinity (Ca^{2+})).

A spatial pattern of the flux is also defined, in the file:

```
biogem_force_flux_ocn_ALK_SUR.dat
```

An example pattern has been set – here, bordering the major tropical coral reefs locations in the Western Pacific. In choosing your regions(s), you might think about mitigating specific ecosystem impacts of ocean acidification, or about the feasibility of transport and proximity to abundant limestone ($CaCO_3$ – the source of lime) and/or energy.

7.2.4 Modifying spatial patterns

The spatial patterns of an applied flux forcing to the ocean can easily be modified. The pattern is specified in a simple ASCII (plain text) file, in the file in the forcing sub-directory ending '`_SUR.dat`'. The file (in this example the default Fe pattern) looks like:

Here: ‘0’s represent land and cannot have a forcing associated with them. ‘0.0’s represent a zero flux to the ocean, and ‘1.0’s the default Southern Ocean forcing pattern. Note that a distinction is made between a ‘0’ and a ‘0.0’ so that you can make out where the continents are and do not necessarily have to count in the i and j grid directions to find a specific location. The grid is the same as you saw previously in tracing ocean circulation, and which numbered the i and j axes if that helps. For the ALK forcing, ‘1.0’s are set off the coast of Australia and SE Asia and in the PO_4 forcing, in the Atlantic.

There is no more to changing the pattern of the flux forcing than simply marking with a ‘1.0’ where you would like the forcing applied, and a ‘0.0’ where it should not be. Note that there should be a single blank line at the bottom of the file. (If you have problems applying a modified spatial pattern – check that this is present.) It is best to keep a copy of the original forcing in case you make a mess of the spatial pattern file, but the original can also be recovered from the code server.

7.3 Further ideas

7.3.1 Impacts to look out for

- Impacts and ecosystems of interest could potentially be ones residing on the ocean floor, such as cold-water (deep water) corals, and are not necessarily planktic (/surface) only.
- Don't forget that different calcifying organisms employ different mineralogies (calcite vs. aragonite), with different saturation states and hence potentially susceptibility to ocean acidification. Hence thresholds of both aragonite and calcite saturation will be relevant, depending on the organism. Depending on the organism, saturation changes occurring in specific regions may much more relevant than a global mean change. Also, it might be the seasonal minimum value reached, rather than annual averaged minimum, that is critical.
- Some of the arguments against some forms of ocean carbon geoengineering concern the potential for adverse impacts on marine organisms (and positive climate feedbacks) induced by decreases in the degree of oxygenation in the ocean, such as expanding and/or intensifying oxygen minimum zones. **muffin** saves 3D fields of O_2 concentrations that can be plotted in slices through the ocean of various orientations.

7.3.2 Other thoughts and suggestions

- If you want to combine forcings, you need to first update the file: `configure_forcings_ocn.dat` – this specifies which ocean flux forcing will be used – simply copy the relevant line from the equivalent file of the forcing to be added. You will also need to copy in the relevant '`_sig.dat`' and '`_SUR.dat`' files. Remember that in the `user-config` file, you will need to set the relevant flux scaling parameter for each different flux in the forcing.
- By default, the CO_2 -climate feedback is 'on':

```
# set climate feedback
ea_36=y
```

Should you want to assess the impacts of geoengineering independently of changes in climate – the option is there. (Note that under some of the high end CO_2 emissions scenarios, there may be a degree of collapse of the AMOC that will presumably affect the patterns of ocean acidification and oxygenation etc.)

- If you are having doubts that your experiment is actually 'doing' anything (different from the control) – remember to create anomaly maps (plots) to look for specific changes in e.g. saturation state, pH, or the water column inventory of anthropogenic CO_2 . Even before this – plot anomalies of the flux you think you have applied, looking specifically at the region you think you have applied it to. For this, **muffin** saves the 3D distributions of dissolved Fe and PO_4 . See Figures below.
- Always be aware of the caveats regarding this specific model (and models in general) – how much does it differ from the 'real world' for the modern ocean, particularly in terms of patterns of carbonate saturation? Does it even simulate anthropogenic CO_2 uptake adequately in the first place?

7.3.3 Further modifications of the biological pump in the ocean

Other manipulations of the biological pump and ocean carbon cycle are possible and potentially instructive and in the following examples may be of rather more relevance to past climates and carbon cycles and e.g. possible reasons for the low atmospheric CO_2 concentrations at the last

glacial, as opposed to relevant to geoengineering (a good thing!). The first two of these may have profound effects not only atmospheric pCO_2 but also on dissolved oxygen concentrations in the ocean (and hence implications for the suitability of animal habitat such as for fish) and this is something that you will want to look at as part of your overall assessment of impacts.

Remineralization depth

In the model configuration that you have been using, the degradation of particulate organic matter sinking in the water column proceeds according to a fixed profile of flux with depth (there is no e.g. temperature control on the rate of bacterial degradation of sinking organic matter) with CO_2 and PO_4 released back to the seawater as the particulate flux decreases. The parameter that controls the (e -folding) depth scale of particulate organic matter is:

```
bg_par_bio_remin_POC_eL1=589.9451
```

Either edit this value (found under the heading: # -- REMINERALIZATION --) or add a new line at the end of the *user-config* file specifying the value you want. Units are m .

Read *Ridgwell et al.* [2007] for additional discussion of this parameter. See Figure 2-4 in *Ridgwell* [2001] (http://www.seao2.org/pubs/ridgwell_thesis.pdf) for an illustration of how the flux of particulate organic matter decreases with depth in the ocean, plus references therein.

There is also an associated parameter: `bg_par_bio_remin_POC_frac2`, which sets a fraction of organic matter that is assumed to settle through the water column completely un-altered (currently assigned a value of 0.045 == 4.5%), but this is arguably less useful to change than the remineralization length-scale of the more labile fraction (the other 95.5% of particulate organic carbon exported from the ocean surface).

Note that there may well be no simple parallel that can be found in geoengineering to this process. However, there are hypotheses that during the last glacial and as a result of colder ocean temperatures, the depth scale was longer. Conversely, there are ideas about that the warmer temperatures of the e.g. Eocene ocean and hence faster rates of bacterial metabolism led to a much shallower remineralization depth scale. So a remineralization depth scale that is responsive to temperature may have importance in understanding ocean biogeochemical cycles during both past warm and cold climates as well as obviously, future global change. While you are not implementing a temperature-dependent parameterization explicitly, you can at least test for whether changes in temperature might have important impacts by simply changing the remineralization depth to be shallower (smaller depth-scale under a warming climate) or deeper (greater depth-scale in a colder ocean).

Macro nutrient inventory and uptake

Suggestions have been made that nutrients were used more efficiently during the LGM, meaning that for the same nutrient uptake at the surface more carbon was exported to depth in the ocean. See: *Omta et al.* [2006]. There are also a bunch of (relatively old) hypotheses concerning differences between glacial and modern ocean in how much nitrate (NO_3^-) there was. There is no NO_3^- in this version of **muffin** (just PO_4 and Fe), but an analogous change can be made to the phosphorous cycle.

For the nutrient-to-carbon ratio in organic matter, the relevant parameter is:

```
bg_par_bio_red_POP_POC=106.0
```

To change the default value (106.0), add a new line at the end of the *user-config* file specifying the value you want. A larger number means that PO_4 is being utilized more efficiently and more organic matter is being produced for the same nutrient consumption.

To test the effect of there being more PO_4 in the ocean, in addition to using the (surface) flux forcing as described earlier, it is also possible to simply increase the inventory of the ocean as a whole in one go:

```
bg_ocn_dinit_8=1.0E-6
```

which will add $1 \mu mol kg^{-1}$ of PO_4 uniformly to the ocean. (A larger/smaller number will obviously increase the glacial nutrient inventory by more/less.)

In terms of geoengineering, changing the ‘Redfield’ ocean plankton might be difficult ... but not impossible, although we are presumably talking about releases of genetically modified organisms to the entire ocean to achieve this meaning there are obviously some severe ethical concerns. However, adding macro nutrients such as PO_4 (more often, NO_3^- is talked about) may be more feasible.

CaCO₃:POC rain ratio

Kicked off by a classic 1994 *Nature* paper by *Archer and Maier-Reimer* (see: *Kohfeld and Ridgwell* [2009]), one potential means of changing atmospheric CO_2 naturally at the last glacial involves changes in the export ratio between $CaCO_3$ (shells) and POC (particulate organic matter). Such a change in ratio could come about through a variety of ways (e.g., via the ‘silica leakage hypothesis’ (see: *Kohfeld and Ridgwell* [2009]) and also through the direct effect of Fe on diatom physiology (see *Watson et al.* [2000] in *Nature* and also Supplemental Information). There are also ideas about an opposite ocean acidification effect, whereby the less acidic glacial (compared to modern) ocean led to increased calcification and $CaCO_3$ export. Note that this response (higher saturation == greater rate of calcification) is encoded into your model configuration – see *Ridgwell et al.* [2007b].

In **muffin**, the $CaCO_3 : POC$ rain ratio is controlled (technically: scaled) by the parameter:

```
bg_par_bio_red_POC_CaCO3=0.0485
```

The pattern of $CaCO_3 : POC$ rain ratio is not uniform across the ocean (why? (see: *Ridgwell et al.* [2007, 2009])), and its pattern can be viewed in the (2D **BIOGEM**) netCDF variable: misc_sur_rCaCO3toPOC.

(Note that it is unlikely that there is any parallel in a geoengineering context to this process.)

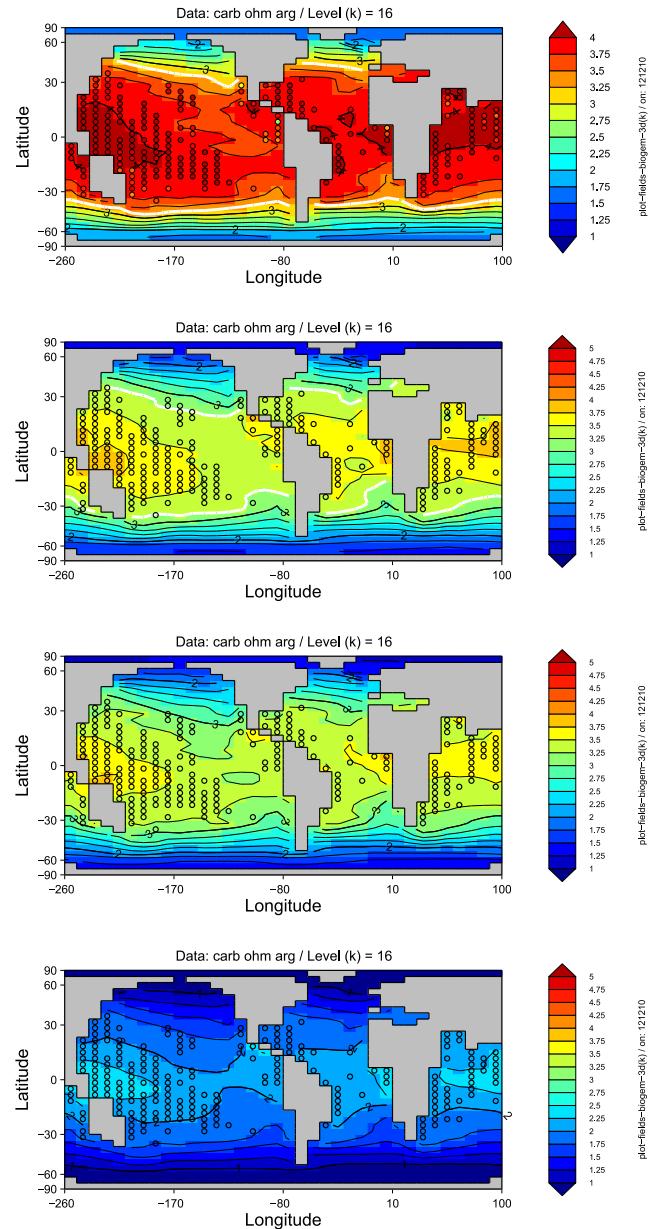


Figure 7.1: Mean annual ocean surface saturation (aragonite) changes. Top: pre-industrial model ocean surface saturation (aragonite) with ReefBase tropical coral reef locations re-gridded to the **muffin** grid and color-coded with modern observationally-based saturation values. 2nd and 3rd down: Year 1994 and 2010 ocean surface saturation (aragonite) with ReefBase reef locations. Bottom: Year 2010 ocean surface saturation (aragonite) under the A2 CO₂ emissions scenario. The thick white line delineates the 3.25 saturation contour (inferred to reflect a limitation on corals). Examples here produced using **muffinplot** but equally do-able in **Panoply** with the exception of achieving a data overlay. These are provided simply to illustrate some of the impacts you might consider and possible ways of visualizing them.

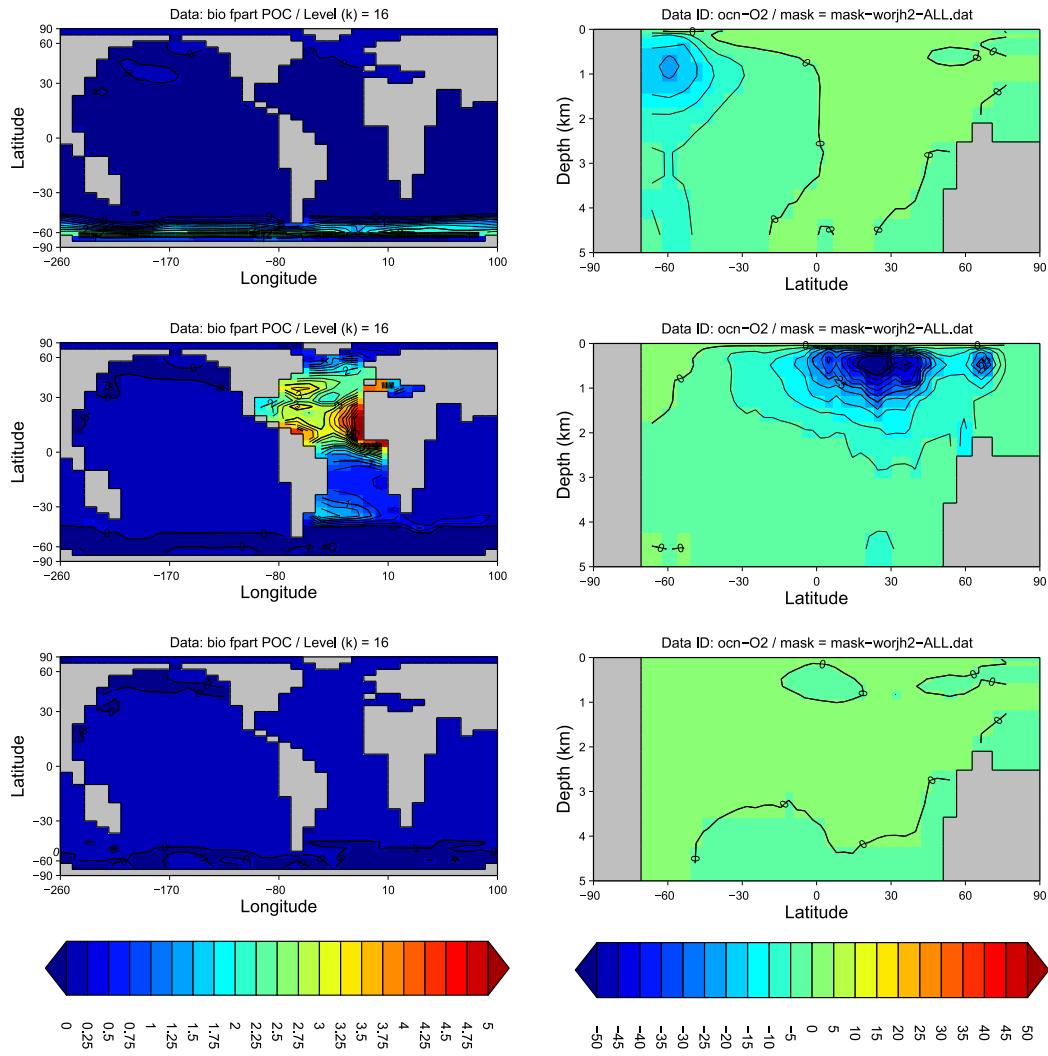


Figure 7.2: Ocean surface export (particulate organic carbon) and zonal $[O_2]$ anomalies. Left: anomalies of global mean annual export production, for Fe fertilization (top), PO_4 addition (middle), and ocean liming (bottom). Right: Zonal mean anomalies of dissolved O_2 concentrations. Examples here produced using **muffinplot** but equally do-able in **Panoply** with the exception of achieving a data overlay. These are provided simply to illustrate some of the impacts you might consider and possible ways of visualizing them.

Getting going with ECOGEM

- Running the model
- Viewing 2D time-slice output
- Comparing to observations

Ecosystem configuration

- Visualising composite data
- Iron limitation

Increasing ecological complexity

- Plankton size classes
- Viewing 2D time-slice output
- Create your own ecosystem

Build it up, tear it down

- A fully size-structured ecosystem
- Ecosystem characteristics
- Mixotrophy

Ecology in past oceans

Ecology in fake oceans

In the chapter we address the role and nature of marine (plankton) ecosystems.¹

muffin includes an explicit ecosystem component, including primary and export production as well as plankton biomass – **ECOGEM**² – as an alternative option to the ‘biologically induced export flux’ representation of export production. The ecological model takes what is known as a size-structured approach to representing diversity of function in marine ecosystems, and is flexible in being able to be configured to represent any range of size classes of phytoplankton and zooplankton (and/or mixotrophs).

Stuff to keep in mind...

- We will be working with highly idealised ecosystems in a highly idealised ocean.
- The aim is to explore why the model behaves as it does.
- The assumption is that this will give us some insight into why the real world behaves as it does. Perhaps. (It is up to you to question the validity of this assumption.)

¹Loosely based on original workshop material devised by Ben Ward <b.a.ward@soton.ac.uk>

²Ward et al. [2017] – Ward, B. A., Wilson, J. D., Death, R. M., Monteiro, F. M., Yool, A., and Ridgwell, A.: EcoGENIE 0.1: Plankton Ecology in the cGENIE Earth system model, *Geosci. Model Dev. Discuss.*, <https://doi.org/10.5194/gmd-2017-258>, 2017.

Read.me

Before you start, you need to retrieve some a **muffin** model experiment *re-start* files. At the terminal, enter the following commands (hitting ‘return’ after each command):

```
cd ~/cgenie_output  
wget http://www.seao2.info/cgenie_output/_restarttest.worlg4.Wardetal20018.ECOGEM.SPIN.tar.gz  
tar xfzv _restarttest.worlg4.Wardetal20018.ECOGEM.SPIN.tar.gz
```

Then as before ... return to the **muffin** genie-main directory:

```
cd genie-main
```

8.1 Getting going with ECOGEM

Previously, you were running the standard 'biogeochemical' version of **muffin**³. In **BIOGEM**, the biological pump is driven by an implicit (i.e. unresolved) biological community. As in the real ecosystem, the biological uptake of carbon and nutrients (such as phosphorus and iron) is limited by light, temperature and nutrient availability. However, unlike the real ecosystem, any uptake is *directly* and *instantly* converted to particulate and dissolved organic matter (POM, DOM) and exported to the ocean interior via (gravitational) settling and the ocean surface, respectively. i.e.

- surface inorganic nutrients $\xrightarrow[\text{and export}]{\text{production}}$ POM and DOM

POM is then instantaneously⁴ remineralized in the ocean interior, returning nutrients and dissolved carbon and consuming oxygen (amongst other transformations and depending on the specific tracers selected in the ocean). DOM, released in at the ocean surface, is transported with the circulation of the ocean but its fast decay and remineralization time-scale ensures that nutrients are primarily returned back to the surface ocean.

In contrast, in this chapter, we are going to get started with the '**ECOGEM**' ecological modeling package. This will allow us to extend the capabilities of **muffin** to examine a range of questions relating to the role of physiology and community structure in regulating the biological pump and hence atmospheric CO_2 etc. In **ECOGEM**, biological uptake is again limited by light, temperature and nutrient availability, but here it must pass through an explicit and dynamic intermediary plankton biomass pool, before being expressed as the production of POM and DOM (whose fate proceeds as per above):

- surface inorganic nutrients $\xrightarrow{\text{production}}$ plankton biomass $\xrightarrow{\text{export}}$ POM and DOM

8.1.1 Running the model

We will start with the simplest possible configuration of **ECOGEM**, with just a single (small) phytoplankton class⁵. You can run this model at the command line, by entering the following command (note that this should be one continuous line) ...

```
$ ./runmuffin.sh muffin.CBE.worlg4.BASESFeTDTL LABS wardetal.2018.ECOGEM.EXAMPLE 10
_restarttest.worlg4.Wardetal20018.ECOGEM.SPIN
```

As before, we have five input parameters after the run script (`./runmuffin.sh`) is invoked:

1. `muffin.CBE.worlg4.BASESFeTDTL` → The '*base-config*' file.
Note the additional '`_eg`' that marks the inclusion of the **ECOGEM** package.
2. `LABS` → As before – the *user-config* directory where the *user-config* file resides.
3. `wardetal.2018.ECOGEM.EXAMPLE` → The specific *user-config* file (experiment name) used in this example.
4. `10` → The run duration (for this particular example experiment) ... in years.

³e.g. see *Ridgwell et al. [2007]*

⁴Depending on the specific remineralization options chosen. The default is in fact for POM to settle downwards at a finite rate.

⁵i.e. as if the ocean was populated with a small species of phytoplankton (photo-autotroph) and nothing else.

5. `_restarttest.worlg4.Wardetal20018.ECOGEM.SPIN` → The *restart* filename.

Here we are telling the model to start from the endpoint of a previous experiment (a 10,000 year run of **BIOGEM** ... although what you actually have, is a 10 year run following on from a 10,000 year run ...).

Note that the *re-start* here only applies to the biogeochemical part of the model. The ecological community will be initialised from some very low biomass.

The model will run as before, except that now we have the **ECOGEM** debug option enabled, and you will get a lot of extra information about how the ecological model is configured.

The additional lines appearing as the experiment runs, e.g.

```
>>> SAVING ECOGEM TIME-SLICE AVERAGE CENTERED @ year : 0.500
```

confirm that **ECOGEM** is writing ecological time-slices at the same time as **BIOGEM** is writing its own time slices. (Note that **ECOGEM** is not currently set up to save *time-series* data in the same way as **BIOGEM**.)

8.1.2 Viewing 2D time-slice output

Following the same convention as for **BIOGEM**, **ECOGEM** *time-slice* output is saved in the sub-directory of your experiment results directory, named **ECOGEM**.⁶

1. Open the `fields_ecogem_2D.nc` file by locating it in the correct directory, and double clicking on it in the file transfer window (if you have that software function configured), or transfer locally and then open (e.g. in **Panoply**).
2. You should now see a list of 2D arrays that were output by **ECOGEM**. Looking at the Long Name description, simply click on a variable of interest. If a menu window pops up, just click on Create or hit the Return key.
3. Check the **Panoply** settings to make sure you really know what you are looking at.
 - (a) Which *time-slice* (i.e. simulation year) are you looking at?
 - (b) What is the data range (i.e. colour scale)

Remember: you can change the default settings in **Panoply** to avoid changing things every time you open a new file. First click on **Panoply** → Preferences.... Here you can switch off interpolation and the grid overlay under the General menu. You can disable the spuriously precise coastline under Lon-Lat Plots.

A guide to some key **ECOGEM** output variables to peruse can be found at the end of Chapter 15.

8.1.3 Comparing to observations

Models are usually intended as an approximation of the real world (whatever that is). It might, therefore, be useful to check if our approximation is in anyway realistic. We can do this by comparing the model output to observations.

1. You can download a compilation of key biogeochemical variables (as netCDF files) from the mymuffin [webpage](#).

⁶for this particular example experiment, the full path to the **ECOGEM** results would be:
`/cogenie_output/Wardetal.2018.ECOGEM.EXAMPLE/ECOGEM/`.

2. You can open the GEnIE_observations.nc file in **Panoply** in the same way as you opened the fields_ecogem_2D.nc file.
3. You can now compare the model output to key biogeochemical variables, such as surface chlorophyll or phosphate. (Also, e.g. create difference maps.)
4. When doing this, be asking yourself the question: doe the model perform well or poorly with respect to reproducing these variables? If not, why not?

NOTE: **ECOGEM** only saves a limited number of surface (2D) data arrays. You can look at other variables (in 2D and 3D) by opening the corresponding **BIOGEM** files⁷, fields_biogem_2D.nc and fields_biogem_3D.nc in **Panoply**.

Bear in mind that we restarted the model from a previous simulation with **BIOGEM**. Some of the state variables may take a long time (i.e. $\gg 10$ years) to adjust to the new model configuration (especially the distribution of nutrients and oxygen at depth).

⁷They can be found in the directory:
~/cgenie_output/wardetal.2018.ECOGEM.EXAMPLE/BIOGEM/

8.2 Ecosystem configuration

In the last section you ran a very simple configuration of the **ECOGEM** ecosystem model, and compared it to observations. In this section we are going to add a bit more ecological realism, with the aim of improving model performance (i.e. as contrasted against observations). We will start by adding a zooplankton population that should bring a degree of ‘top-down’ control to the phytoplankton population.

Details of the original (full) ecosystem are specified in the *user-config* file:

wardetal.2018.ECOGEM.EXAMPLE

1. Locate the *user-config* file (in `~/cgenie.muffin/genie-userconfigs/LABS`), and open it in your preferred text editor.
2. The *user-config* file can be used to configure the model to your liking. One of the most important amendments to note straight away can be seen on line 11, `bg_par_bio_prodopt="NONE"`. This effectively disables the simple biological export scheme in **BIOGEM**, replacing it with the explicit biology of **ECOGEM**. This is a necessary step whenever running **ECOGEM**, because we do not want the implicit and explicit biological schemes to be implemented simultaneously ...
3. We can also see a load of other model parameters. Any that begin with ‘`bg_`’ correspond to **BIOGEM**, while ‘`eg_`’ corresponds to **ECOGEM**. The **ECOGEM** parameters begin after line 71.
4. One of the most important parameters specifies the *ecosystem configuration* file:

```
eg_par_ecogem_plankton_file ='NPD.eco'
```

This points to a file (located in `~/cgenie.muffin/genie-ecogem/data/input/`) that specifies every plankton population (‘species’, if you like) that is accounted for in the model experiment. If you open that file in a text editor, you will see something akin to the following:

```
01          02      03
\|          \|      \|
-START-OF-DATA-
Phytoplankton    10.00   1
-END-OF-DATA-

\|          \|      \|
01          02      03

DATA FORMAT AND ORDER
-----
COLUMN #01: plankton functional type name
COLUMN #02: plankton diameter (micrometers)
COLUMN #03: number of randomised replicates

INFO: TRACER ASSIGNMENT RULES
-----
Plankton functional type one of: Prochlorococcus
                                Synechococcus
                                Picoeukaryote
                                Diatom
                                Coccolithophore
                                Diazotroph
                                Phytoplankton
                                Zooplankton
                                Mixotroph
```

5. The first thing to note is that only the lines in between the -START-OF-DATA- and -END-OF-DATA- tags are read by the computer. The rest is there solely for your guidance.

Each line that is entered in the computer-readable area tells the model to create a distinct plankton population in the model. The ‘plankton functional type’ of this population is specified in the first column, while the plankton diameter is specified in the second column. A ‘1’ must always be placed in the third column (it doesn’t ‘do’ anything, but the model still needs it ... why ... ?).

In this ‘NPD’ (single nutrient-plankton-detritus) configuration, we only have a 10 micron generic phytoplankton. The ecological and physiological traits of this population are assigned automatically according to the size and the functional type (here: photo-autotroph).

NOTE: The only PFTs available at the moment are Phytoplankton, Zooplankton and Mixotroph. The other groups currently have no real functionality associated with them.

NOTE: This file format is fussy, and you cannot have any empty lines in between the -START-OF-DATA- and -END-OF-DATA- tags – every line between these tags must have data (3 parameter values).

6. We can increase the ecological complexity of the model by adding another plankton population. Save the *ecosystem configuration* file under a new and highly intuitive name (such as NPZD.eco), and add another line specifying a 100 micron zooplankton. It is important that the zooplankton is 10 times larger than the phytoplankton in terms of diameter. This is the optimal predator-prey length ratio in the default configuration. (You could maybe think about changing this value later on.)

7. To run the model with this new configuration, change the name of the *ecosystem configuration file* in the *user-config* file...

```
eg_par_ecogem_plankton_file ='NPZD.eco'
```

8. Save the new *user-config* file under a different name (e.g. BSS.NPZD.SPIN). You can now execute the model at the command line⁸ ...

```
./muffin.CBE.worlg4.BASESFeTDTL / BSS.NPZD.SPIN 10
    _restarttest.worlg4.Wardetal20018.ECOGEM.SPIN
```

... or submit it to the cluster queue ...

```
qsub -q dog.q -j y -o cgenie_log -V -S /bin/bash runmuffin.sh
muffin.CBE.worlg4.BASESFeTDTL / BSS.NPZD.SPIN 10
    _restarttest.worlg4.Wardetal20018.ECOGEM.SPIN
```

9. Once you have completed the new simulation, compare the new results to the old simulation, and also in terms of its ability to reproduce observations. Has the addition of zooplankton to the model improved its behaviour or not? Look also at the global distributions of carbon biomass in the phytoplankton and zooplankton populations (again, a log scale might help).

How have the zooplankton interacted with the phytoplankton to change the ecological dynamics in the model?

8.2.1 Visualising composite data

We can perhaps get a better handle on this question by looking at the ratio of phytoplankton-to-zooplankton biomass. Such ratios can, however, be difficult to assess simply by eye-balling two maps. Instead we can use **Panoply** to combine data arrays.

⁸Don’t forget to change the name of the *user-config* file here as well ...

1. First close all your **Panoply** plot windows. Then open a new one for C Biomass - Popn. 001 (10.00 micron phytoplankton). Next, select C Biomass - Popn. 002 (100.00 micron zooplankton), and click the Combine Plot icon at the top of the **Panoply** window.
2. A box will open up asking you In which existing plot should I combine the variable. As you now only have one plot available, this should be a straightforward choice. Click Combine.
3. A new map should appear showing the total zooplankton carbon biomass minus the total phytoplankton carbon biomass (see the label on the colour scale). This is not what we want. Below the map, under the Array(s) tab, there is a drop down menu showing the range of different ways the two arrays can be combined. We want to look at the Z:P biomass ratio, so select Array 2 / Array 1.
4. You now need to make sure that you are looking at the right year (you can time-lock the two arrays by clicking on the chain icon). You may also find it helpful to look at the data on a log scale, with a scale range of 0.1 to 10. You might also like to change the Color Table: option to GMT_polar.cpt.

Questions:

- What does this plot say about the relationship of zooplankton and phytoplankton in different regions of the ocean?
- In what regions do zooplankton or phytoplankton dominate?
- What affect does a high Z:P ratio have on the biomass of the phytoplankton population?⁹

8.2.2 Iron limitation

Up to this point, we have only considered phosphate as a limiting nutrient. (Iron was included in the model, but it was not limiting to phytoplankton growth.) You can switch on iron limitation by modifying two lines in the *user-config* file:

`eg_useFe=.true.`

and

`eg_fquota=.true.`

Give the *user-config* file a new name (e.g. BSS.NPZD_Fe.SPIN), then re-run the model.

1. Examine the effect of iron limitation in the new model. What has changed?
2. We can get a more exact picture of the nutrient limitation terms via the netCDF output variables: `eco2D_xGamma_Fe_001` and `eco2D_xGamma_P_001`. These two variables take values of between 0 and 1. A 1 indicates that the factor is not limiting to growth. A 0 indicates the factor is completely preventing growth.
 - In what regions are iron and phosphorus more or less limiting to growth?
 - In regions where neither is limiting, what other factors might be important?
3. Plankton stoichiometry plays a critical role in determining which nutrient is most limiting to growth. You can increase the plankton $Fe : C$ ratio by increasing the minimum and maximum iron quotas. Look at the parameters `eg_qminFe_a` and `eg_qmaxFe_a` in the *user-config* file.

⁹For example, in terms of the chlorophyll concentration.

- What happens to the ecosystem if you increase these parameters by a factor of 2, 5 or 10?
- How does a change in these parameters affect the model behaviour?
- What has changed in terms of the patterns of nutrient limitation?
- What has happened to the concentration of the limiting and non-limiting nutrient?

NOTE: It can be 'risky' to just change the parameter value in place, as you might forget what you started with ... Instead; you might copy/paste a new version of the line in question, and comment out the original by placing a '#' at the beginning of the line. For example:

```
#eg_qminFe_a = 3.0e-6  
eg_qminFe_a = 6.0e-6
```

changes the minimum iron quota by a factor of 2, whilst keeping a record of the original setting (inactivated by the #). Or, you might do something like:

```
eg_qminFe_a = 6.0e-6 # 3.0e-6
```

that reduces the total number of lines you end up with in the user-config file.

4. Nutrient supply ratios are also important in determining the limiting nutrient.
The `bg_par_det_Fe_sol_exp` parameter determines the solubility of atmospheric iron inputs in seawater. Decreasing the value of `bg_par_det_Fe_sol_exp` will therefore decrease the iron-to-phosphorus supply ratio.
 - What happens to the ecosystem if you decrease `bg_par_det_Fe_sol_exp` by 10, 20 or 50%?

8.3 Increasing ecological complexity

In the last section, we looked at the results of some simulations based on ‘NPD’ (one nutrient-(phyto)plankton-detritus) and ‘NPZD’ (one nutrient-(phyto)plankton-zooplankton-detritus) type ecosystem models. Here we will begin to incorporate a bit more ecological complexity.

8.3.1 Plankton size classes

We are going to add a few more plankton size classes, so we have small, medium and large phytoplankton and zooplankton.

1. Save the *ecosystem configuration* file under a new name (e.g. 3P3Z.eco), replacing the existing plankton populations with the ones described in Table 8.1.
2. To run the model with this new configuration, change the name of the *ecosystem configuration file* in the *user-config* file:

```
eg_par_ecogem_plankton_file='3P3Z.eco'
```

3. Save the new *user-config* file under a different name (e.g. BSS.3P3Z.SPIN) and then run the new model at the command line (e.g. for 10 years).

Table 8.1: Plankton functional groups and sizes.

<i>j</i>	PFT	Diameter (μm)	<i>j</i>	Functional Type	Diameter (μm)
1	Phytoplankton	0.6	4	Zooplankton	6.0
2	Phytoplankton	6.0	5	Zooplankton	60.0
3	Phytoplankton	60.0	6	Zooplankton	600.0

8.3.2 Viewing 2D time-slice output

Open up the 2D *time-slice* data for the new experiment, following the same procedure as in the previous section. You will now see a lot more *time-slice* variables now appear listed in **Panoply**. We have all the same diagnostics as before, plus some new ones relating to the new plankton populations you have just added. We also have a load of other arrays describing the size distribution and diversity of the photosynthetic community (non-phototrophic populations are ignored in these metrics). These were not included before, because there was only one phytoplankton population.

Size fractions

Variables beginning “eco2D_Size_Frac_...” give the chlorophyll biomass in the three size fractions:

1. picophytoplankton (diameter $\leq 2 \mu\text{m}$)
2. nanophytoplankton ($2 < \text{diameter} \leq 20 \mu\text{m}$)
3. microphytoplankton (diameter $> 20 \mu\text{m}$)

Size metrics

Any other variables beginning ‘eco2D_Size_...’ give metrics describing the phytoplankton size distribution.

- eco2D_Size_Mean: Geometric mean¹⁰ phytoplankton diameter, weighted by carbon biomass

¹⁰ We use the geometric mean and standard deviation, because phytoplankton biomass is approximately log-normally distributed across the phytoplankton size range.

- eco2D_Size_Stdev: Geometric standard deviation¹¹ of phytoplankton diameter, weighted by carbon biomass.
- eco2D_Size_Minimum: Diameter of smallest phytoplankton contributing >0.1% of the total phytoplankton carbon biomass.
- eco2D_Size_Maximum: Diameter of largest phytoplankton contributing >0.1% of the total phytoplankton carbon biomass.

Diversity metrics

Any variables beginning 'eco2D_Diversity...' give metrics describing the phytoplankton diversity.

- eco2D_Diversity_Threshold: the threshold diversity index. The number of species contributing >0.1% of the total phytoplankton carbon biomass [Barton et al., 2010]¹².
- eco2D_Diversity_Berger: the inverse Berger(-Parker) index [Berger and Parker, 1970]¹³. The proportion of carbon biomass made up by all but the single most dominant population. For example, if the dominant population accounts for 40% of the total carbon biomass, inverse Berger (-Parker) index is 0.6.
- eco2D_Diversity_Simpson: the inverse (Gini-)Simpson index [Simpson, 1949]¹⁴. This is effectively the probability that two samples taken at random from the community will be from a different species (note that the probability of selecting a population is dependent on carbon biomass, not cell abundance). If we define the proportional biomass of each species as its relative contribution to the total carbon biomass in the community, the inverse Gini(-Simpson) index is calculated as one minus the sum of the squares of the proportional biomasses of each species.
- eco2D_Diversity_Shannon: the Shannon(-Wiener or -Weaver) index [Shannon, 1948]¹⁵. With the proportional biomass defined as above, the Shannon index is defined as the sum of [the proportional biomass multiplied by the logarithm of the proportional biomass] for each species.

NOTE: The threshold index is a fairly crude measure of the total number of species in the community, relative to a small and arbitrary threshold of relative biomass. This index is not very sensitive to the relative biomass of individual species (although one very successful species can raise the absolute value of the threshold, thus lowering the diversity). The other three indices do more to quantify the evenness of the community. The more unequal the proportional abundances, the smaller the value of the index. If almost all the abundance is concentrated into one type and all the other types are very rare, the latter three indices can become very small. A community with fewer species, but with more evenly distributed biomass, may well have higher values for these three diversity indices.

¹¹ The geometric standard deviation describes the *relative* size range of the phytoplankton. For a geometric standard deviation of σ , ~68.2% of the phytoplankton carbon biomass will be in cells no more than σ orders of magnitude smaller or larger than the geometric mean size.

¹² A D Barton, S Dutkiewicz, G Flierl, J Bragg, and M Follows. Patterns of diversity in marine phytoplankton. *Science*, 327(5972):1509–1511, 2010.

¹³ W H Berger and F L Parker. Diversity of planktonic foraminifera in deep-sea sediments. *Science*, 168 (3937):1345–1347, 1970.

¹⁴ E H Simpson. Measurement of diversity. *Nature*, 163:688, 1949.

¹⁵ C E Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(379-423 and 623-656), 1948.

Have a look at some of these metrics, but bear in mind that they summarise the diversity of a phytoplankton community that includes just three species. They are probably not that revealing, so we will come back to them later. Instead, have a look some of the other metrics describing the model ecosystem.

Questions:

- What are the global distributions of the different size classes?
- How do the global biomass distributions compare to variables such as temperature¹⁶, or primary production (Uptake Fluxes C)?
- How does nutrient, light and temperature limitation vary between the size classes?
- Can you account for the distribution biomass between the size classes according to the different limiting factors?

8.3.3 Create your own ecosystem

Save the *ecosystem configuration* file under a new name and add some more plankton populations (whatever and as many as you like¹⁷). Update the **muffin** parameter `eg_par_ecogem_plankton_file` in the *user-config* and save this file under a new name. Run the new model.

Questions:

- How many populations can you get to coexist (i.e. each having a non trivial (>0.1%) biomass)?
- What effect do the new populations have on the community as a whole?
- What effect, if any, do your additions have on the strength of biological export?
(e.g. look at `bio_fpart_POC` in `fields_biogem_3D.nc`.)

¹⁶Ocean temperature is saved in `fields_biogem_3D.nc`

¹⁷Just bear in mind that the more populations you put in, the slower the model will run!

8.4 Build it up, tear it down

8.4.1 A fully size-structured ecosystem

We are now going to switch to a more diverse version of the size-structured ecosystem model. This configuration has 8 size classes of phytoplankton, and 8 size classes of zooplankton, as shown in Table 8.2.

1. Save the *ecosystem configuration* file under a new name, replacing the existing plankton populations with the ones described in Table 8.2.
2. Update the *user-config* to point to the new *ecosystem configuration* file, and save again under a new name. (It is generally a good idea to make a note of the name and goal of each experiment as you set it up.)
3. Run the new model for at least 20 years (this will probably take about 10-15 minutes).

Table 8.2: Plankton functional groups and sizes.

<i>j</i>	PFT	Diameter (μm)	<i>j</i>	Functional Type	Diameter (μm)
1	Phytoplankton	0.2	9	Zooplankton	0.6
2	Phytoplankton	0.6	10	Zooplankton	1.9
3	Phytoplankton	1.9	11	Zooplankton	6.0
4	Phytoplankton	6.0	12	Zooplankton	19.0
5	Phytoplankton	19.0	13	Zooplankton	60.0
6	Phytoplankton	60.0	14	Zooplankton	190.0
7	Phytoplankton	190.0	15	Zooplankton	600.0
8	Phytoplankton	600.0	16	Zooplankton	1900.0

8.4.2 Ecosystem characteristics

We can now begin to look at the size and diversity metrics of the phytoplankton community in a meaningful way.

1. Look first at:
 - (a) the total carbon biomass
 - (b) the carbon uptake flux (i.e. primary production)
 - (c) the geometric mean size

Make sure in each case that you are looking at the last year of model output. You may also find it useful to adjust the colour scale, or to change to a logarithmic colour scale (e.g. try a logarithmic scale from 2 to 20 microns for the geometric mean size).

Looking at the maps, we can perhaps pick out three different “biomes” in terms of their community properties:

- (a) The low-latitude oligotrophic gyres are relatively unproductive, and support some of the lowest annual mean biomass in the surface ocean. In these regions the mean phytoplankton size is very small.
- (b) Subpolar latitudes between 40° and 50° (either N or S) are much more productive, and support very high annual mean biomass. These communities also have the highest mean sizes of any region.
- (c) The polar oceans are also highly productive (except perhaps the high Arctic), and support relatively high annual mean biomass. These communities are made up (in the

model, at least) of slightly smaller phytoplankton than we see in the subpolar regions.

2. What can we find out about the community structure in these regions? Open up some of the other metrics describing the community (standard deviation of size distribution, size fractionation, diversity, limiting factors). What can you find out about the community structure within each region, in terms of coexistence and exclusion?
 - Does the community span a broad or narrow size range?
 - How many size classes are coexisting in each biome?
 - What is the smallest and largest size class in each biome?
 - How much biomass is concentrated in each size fraction (picoplankton, nanoplankton and microplankton)?

Overall – what factors do you think are most important in terms of dictating the global distribution of each size class?

To find out the answers to these questions, you are going to pull the model apart, and then put it back together. At each stage the aim is to bring in a different limiting factor, so that you can see its effect on the model behaviour.

The fundamental niche

The first step is to find out the impact of abiotic factors on the distribution of different phytoplankton sizes. In other words, we need to find out what the distribution of the phytoplankton would be in the absence of any ecological interactions, such as resource competition and predation. This is effectively their ‘fundamental niche’.

The fundamental niche is fairly abstract, and not something that can be measured in the real world. In model world, however, we can get a useful estimate of the fundamental niche by making a few simple changes to the model.

1. First of all, you can remove all predation, simply by removing the zooplankton from the *ecosystem configuration* file. Once again, you will need to save a new and appropriately named *user-config* file.
2. Next, you also need to remove all competition for nutrients and light. This involves tweaking the model equations so that the phytoplankton are not nutrient limited, and do not attenuate light. To do this, all you need is to add the following line to the *user-config* file.

`eg_fundamental = .TRUE.`

3. If you now run the model (just 10 years should do it, in this case), you should have a community of eight phytoplankton size classes that are growing solely as a function of the incoming light and the temperature. This growth will be balanced balanced the basal (i.e. non-grazing) mortality. As there is no feedback between the ecosystem and the environment, populations that can survive will grow exponentially and without limit, potentially reaching astronomical abundance in very little time. Populations that cannot survive will rapidly decline to almost nothing.

The regions in which each plankton shows positive growth defines its fundamental niche. This is a function of abiotic conditions only, and is the absolute limit of its geographical range. Look at the carbon biomass distribution in each size class (set the data range in each case from 0 to 1mmolCm^{-3}).

- How and why does the fundamental niche vary with size?
- Could the limits of the fundamental niche explain some of the patterns seen in the full model?

Resource competition.

The next step is asses the impact of resource competition. We are first going to do this in the absence of any zooplankton grazing.

1. All you need to do at this stage is to re-enable nutrient and light competition. To do this, simply delete ‘eg_fundamental = .TRUE.’ from the *user-config* file (or comment out the line to disable it), and save under a new name. Leave the *ecosystem configuration* file as it is.
2. You should have a community of eight phytoplankton size classes that are competing for nutrients and light, again as a function of temperature. This is a much more realistic simulation, as feedbacks between the ecosystem and the environment serve to limit the size of the phytoplankton populations.

Examine the model to find out:

- What size classes are able to persist when resource competition is enabled?
- Why are different size classes more or less abundant in different areas?
- How does the distribution of each size class compare to the fundamental niche?
- What are the reasons for any differences?

Phytoplankton biogeography at this stage begins to approximate the realised niche, which defines the range of conditions that support a population in the presence of ecological interactions. Note that at this stage, however, we have ignored the effects of any predator-prey interactions, as the zooplankton grazers are still missing.

Resource competition + one generalist zooplankton

The previous simulation is clearly unrealistic (although, hopefully, informative). You are now going to add back in just a single zooplankton class, that grazes equally on all plankton (including itself).

1. Add a 100 micron zooplankton into the *ecosystem configuration* file, and save under a new name. Also update the *user-config* file to reflect the change, and save under a similar name.
2. You need to modify the model so that the zooplankton eats all prey with equal preference. This can be done by adding the following lines to the *ecosystem configuration* file.

```
eg_ns=1  
eg_pp_sig_a=1.0e99
```

NOTE: For aficionados, the first parameter disables prey-switching (i.e. predators no longer preferentially attack the most abundant prey). The second parameter increases the width of the grazing kernel (i.e. predators can attack a range of prey across a huge size range with equal preference).

3. The addition of zooplankton to the model community should give a more accurate approximation of the realised niche.
 - Does the addition of a single zooplankton grazer enable more or less coexistence?
 - What factors might be responsible for any shifts in biogeography?

Resource competition + one “switching” zooplankton

You began with a full food-web containing 8 phytoplankton and 8 zooplankton size classes. The diversity of zooplankton clearly has an effect on the phytoplankton community that is not seen in the previous experiment. This effect can be imitated with just one generalist zooplankton if we instruct it to graze preferentially on the most successful prey.

Re-enable this 'prey switching' effect by changing the following control parameter to a 2:

```
eg_ns=2
eg_pp_sig_a=1.0e99
```

Compare this simulation to the first experiment (8 phytoplankton and 8 zooplankton) to see how the inclusion of prey switching increases coexistence through the 'kill-the-winner' mechanism.

- How does nutrient limitation change with phytoplankton size, and how might zooplankton be affecting this?
- Look at the C:P biomass ratio in the community as a whole, and compare to your estimates from the NPZD model (Lesson 1).
- How does the C:P ratio vary with size? How does having a diverse community affect the coupling of carbon and limiting nutrients?

Further questions to answer

- What sets the fundamental niche, and how does it change with size?
- How is the fundamental niche modified by resource competition?
- What species are favoured in terms of nutrient competition?
- How is the outcome of competition affected by...
 - Abiotic conditions?
 - Increased mortality (through generalist grazing)?
 - Density-dependent mortality (through specialist grazing)?
- Do these experiments tell you all you need to know?
What other modifications can you think of making?

8.4.3 Mixotrophy

Try adding some mixotrophs to the phytoplankton and zooplankton already present in the community. These will have exactly half the nutrient uptake traits of phytoplankton of a similar size, and half the prey capture traits of zooplankton if a similar size. To do this:

1. Save the previous *ecosystem configuration* file under a new name.
2. Edit the new *ecosystem configuration* file and add an additional line to add a mixotroph:

```
Mixotroph    xxx    1
```

where *xxx* is the class size of the mixotroph. (And remember to save it.)

3. Update the *user-config* to point to the new *ecosystem configuration* file, and save again under a new name. (It is generally a good idea to make a note of the name and goal of each experiment as you set it up.)
4. Run the new model for at least 10-20 years.

Further questions to explore/answer:

- How does this effect the mean and standard deviation of cell size?
(Size and diversity metrics will be calculated for phytoplankton and mixotrophs together)
- How does mixotrophy effect the C:P ratio of organic matter?
- How does the realised niches of mixotrophs compare to the fundamental niches of phytoplankton?

Also: try replacing all the phytoplankton and zooplankton with a range of sizes of mixotrophs. How does the simulation differ from one with the same size range of separate phytoplankton and zooplankton classes?

For instance – you might also try having an ecosystem comprising just a single small (e.g. $10\mu m$) mixotroph (no phytoplankton and no zooplankton), and perhaps an ecosystem with one small mixotroph $10\mu m$ and one additional mixotroph, 10 times larger $100\mu m$. Compare the productivity of such an ocean compared to some of your previous simplified ecosystem configurations (such as a single small $10\mu m$ phytoplankton, and a configuration with a single small phytoplankton and a single larger $100\mu m$ grazer).

8.5 Ecology in past oceans

Here we consider a series of examples¹⁸ of the **ECOGEM** model of marine ecology, applied to published paleo configuration of **muffin** and used to ask questions of how different could marine carbon cycling (and atmospheric pCO_2) and oxygenation have been in the (relatively recent) past and how do model projections compare with proxy observations.

The three examples come firstly from the Last Glacial Maximum, with direct comparison being made to post glacial time (here, the late Holocene), with the intention to explore the role of climate cooling, altered ocean circulation, and increased iron supply to the ocean in modifying plankton distributions and size structures. Secondly, from around the time of the Paleocene-Eocene Thermal Maximum (PETM) some 55 Myr ago, now both considering a different paleogeography, a warmer ocean, and then transient warming on top of that. Lastly we turn to marine ecology and carbon cycling in the aftermath of the impact and extinction event at the end of the Cretaceous, some 66 Myr ago, and return to our earlier exploration of what happens if you lose the larger plankton in the ocean.

¹⁸Either published or in the works or existing in some publication fantasy ...

The Last Glacial

The Last Glacial Maximum (LGM) (ca. 19 to 23 ka) was characterized by lower sealevel and higher ocean salinity, colder ocean temperatures, and a reorganized meridional overturning circulation in the Atlantic. The latter 2 changes in particular should have impacted marine ecosystems and indeed, observations suggest range migration (temperature-tracking) and shifts in the zone of highest productivity in the Southern Ocean, amongst other impacts. The aim of this particular investigation, is to assess what these ecological changes are (at least in model world).

Provided is a configuration of LGM ocean circulation that has been tuned to fit observations of benthic carbon isotopes, which provide an observational constrain on large-scale ocean circulation. To this, you'll an ecosystem (in place of the default **BIOGEM** 'induced export' scheme). Also provided is a **muffin** configuration for late Holocene ('HOL') (0-6 ka), created in exactly the same way and also tuned to respective (0-6 ka) benthic carbon isotope data. Configuration HOL provides a point of comparison (or control) for you to compare the ecology in a colder, LGM ocean against.¹⁹

The *user-configs* you need to use, or copy-rename, can be found in the directory:

genie-userconfigs/MS/odalenetal.CP2019

(and you can run your experiments from here²⁰, or better, copy-edit-rename the *user-configs* you run experiments with, from the LABS sub-directory).

Read the *readme.txt* file for instructions for the basic set of command line parameters needed, but remember that you may be running your *user-config* with a different name and likely from a different sub-directory. Spin up the following experiments (and then experiment with them later)²¹:

- (1) *muffin.CB.Glteliaa.BASESFeTDTL_rb base-config*
+ *muffin.CB.Glteliaa.BASESFeTDTL_rb.SPIN user-config*
- (2) *muffin.CB.Glteliiva.BASESFeTDTL_rb base-config*
+ *muffin.CB.Glteliiva.BASESFeTDTL_rb.SPIN user-config*

Submit these to the cluster ... remembering that you will need to recompile **muffin** between (1) and (2) (and potentially also re-compile before running (1) if you have not used that particular *base-config* immediately prior). Run the spin-ups for 10,000 years.

Neither of these configurations currently have an ecosystem enabled, but they will provide a baseline against which you can contrast a pair of experiment that include explicit ecology. What you need to do know is to add in/enable **ECOGEM**. To do this, you need to modify both *base-* and *user-config* files of both the HOL and LGM model configurations.²²

1. *base-config* – First, you need to enable the **ECOGEM** module.

At the top of the HOL *base-config* file *muffin.CB.Glteliaa.BASESFeTDTL_Crb*²³ you will see the line:

ma_flag_ecogem=.FALSE.

Simply change this to *.TRUE.*

¹⁹i.e. you run pairs of HOL and LGM experiments and contrast between them, rather than necessarily comparing to previous modern configurations.

²⁰If you use them in this directory, make sure at the command line, you replace LABS with MS/odalenetal.CP.2019.

²¹Noting that the long file-names differ only in a single 'v' vs. an 'a' ...

²²Strongly recommended that that you copy-rename both sets of files and edit the copies.

²³And similarly for the LGM one.

2. *user-config* – Next, some deletions and additions are needed in the *user-config* provided (which only has implicit biological export enabled).

In the section of the file under the heading:

```
#  
# --- BIOLOGICAL NEW PRODUCTION -----  
#
```

you are going to delete everything there (in that section) and replace it with:

```
# biological scheme ID string  
bg_par_bio_prodopt="NONE"
```

The only other thing you need then is a section of code that defines all the **ECOGEM** ecological parameters.

Go open file wardetal.2018.ECOGEM.SPIN, which you can find in the *user-config* sub-directory MS/wardetal.2018, and you'll see under the heading:

```
#  
# --- ECOGEM -----  
#
```

a long list of parameter settings (down to the next section headed DATA SAVING). Simply copy-paste this entire section (including ECOGEM header lines if you like), anywhere in your *user-config* file. At the very end of the file would do just fine.²⁴

And lastly ... under:

```
#  
# --- MISC -----  
#
```

(and the end of the section) add the following lines

```
# kraus-turner mixed layer scheme on (1) or off (0)  
go_imld = 1
```

This enables a 'mixed layer depth' scheme in the ocean circulation model that **ECOGEM** needs to calculate light limitation during esp. intervals of high latitude / wintertime deep mixing.

This ... should do-it, i.e. you have added the same tuned ecosystem model as described in Ward *et al.* [2018] to your LGM / HOL *user-configs*.

Now you are ready to run new LGM and HOL experiments, with a full ecosystem in each²⁵.

Obvious questions to investigate include not only how ecosystems and patterns of biological export may have differed between LGM and HOL, but also how patterns of nutrients (PO_4 and Fe) may have differed ... and also distributions of dissolved O_2 in the ocean (and in the interior, rather than across the ocean surface). A water mass ventilation age tracer has also been simulated and the results of this will help in understanding how global circulation patterns differ between the 2 time intervals.

You can also compare between with and without **ECOGEM** versions as the only thing that changes between pairs of HOL or pairs of LGM experiments is the biological scheme²⁶.

²⁴Try and ensure that there is blank line at the very end of the file just in case **muffin** has any problems reading it in.

²⁵Remembering that presumably both your *base-config* and *user-config* file names are different as compared to running the non **ECOGEM** version described in the *readme*.

²⁶Actually, this is not true, as in the **ECOGEM** enabled experiments, the mixed layer scheme in the ocean circulation model is also activated and which has an impact on ocean circulation. You could then create a 3rd set of HOL+LGM experiments, using the basic **BIOGEM** implicit export biological scheme, but now also setting the *go_imld* parameter to a value of 1

The PETM

In this practical we are going to look at the ocean as it *might* have been just over 55 million years ago, at the Paleocene-Eocene Thermal Maximum (PETM) – in short a lot warmer, and with a somewhat different continental configuration and hence ocean circulation. The exercise is based on a recent 'ECOGENIE' (**muffin** configured to include **ECOGEM**) publication²⁷ which you should read first.

We are going to make the rather strong assumption that the ecosystem is structured according to exactly the same rules as in the modern ocean, and simply run the same ecological model configuration but in a new climate and ocean environment. However, because we don't really have any good (or any!) data constraints on the iron supply to the early Eocene ocean via e.g. dust, the ecological configuration does not include iron as a limiting nutrient. Bear that in mind when thinking about your results.

The *user-configs* you need to use, or copy-edit-rename, can be found in the directory:
genie-userconfigs/MS/wilsonetal.2018
(and you can run your experiments from here, or better, copy-edit-rename the *user-configs* you run experiments with, from the LABS sub-directory).

Read the *readme.txt* file for instructions for the basic set of command line parameters needed, but note that you may be running a *user-config* with a different name and likely from a different sub-directory. You want to spin up the following experiments first (and then experiment with them later):

- (1) Modern
- (3) Early Eocene CO₂ and Climate

Submit these to the cluster, but remember that you will need to recompile *muffin* between (1) and (3) (and re-compile before (1) if you have not been using the *base-config* *muffin.CBE.worjh2.BASES* immediately prior to this). Run the spin-ups for 10,000 years.

When you have completed the pair of spin-ups, see what you can diagnose and learn about how ecosystems and the spatial pattern of ecology and biological export differ between a colder modern ocean and the warmer Eocene ocean.

For example, in the warmer Eocene world:

- What has happened to the mean plankton size in different regions?
- What has happened to the fundamental niches in different size classes?
- What has happened to the realised niches?
- Is the system more or less productive?
- Has carbon export gone up or down?

See what you can find out about the two systems and think about the mechanisms that might be responsible for the differences ...

²⁷Wilson, J.D., F.M. Monteiro, D.N. Schmidt, B.A. Ward, and A. Ridgwell, Linking marine plankton ecosystems and climate: A new modeling approach to the warm early Eocene climate, *Paleoceanography and Paleoclimatology*, 33, 1439–1452, DOI: 10.1029/2018PA003374 (2018).

Note that to be more comparable with the Eocene, this particular modern configuration also lacks iron co-limitation. We could also try and remove the effect of the Eocene being warmer than the modern so as to concentrate just on the effect of a different continental configuration and hence ocean circulation. The experiment (included in the *user-config* sub-directory and briefly described in the *readme* file:

(2) Late Paleocene Early Eocene Paleogeography

does exactly this, i.e. attempts to 'remove' the effect of higher ocean temperatures by running an Eocene experiment at $\times 1CO_2$.

Conversely, we could run modern at $\times 3CO_2$ and then compare to the $\times 3CO_2$ Eocene experiment. This alternative comparative experiment is provided as:

(S1) Modern with 3 x CO₂

and will also require a 10,000 year long spin-up ... Note that the 2 strategies, while slightly different, are attempting the same thing (i.e. isolating the effect of paleography and ocean circulation from coeval climate change and warming).

Finally – having run some or all of these spin-ups and contrasted the results (focussing on ecological patterns and biological export, but remembering also to assess differences in ocean circulation), you can investigate the impact of geologically rapid warming a-la the PETM, on the system (esp. ecological patterns and biological export plus ocean circulation). There are two immediately obvious possible approaches:

1. You could use a $\times 1CO_2$ spin-up as a re-start – either or both of modern and Eocene continental configurations – and run the $\times 3CO_2$ experiment from this.

This will give you an instantaneous warming – much faster than occurred associated with PETM onset, and also faster even than modern anthropogenic warming. However, it does provide a nice simple idealized perturbation to investigate and analyse.

An experiment duration of 100, or even 10 years, might be sufficient.²⁸

Conversely and for fun, you could also start from a $\times 3CO_2$ spin-up, and run a $\times 1CO_2$ experiment, to achieve a rapid cooling. Investigate the differential ecological response to rapid cooling vs. rapid warming.

2. Secondly, in the *user-configs*, you might note near the bottom is a *forcing* that determines the value of atmospheric CO₂:

```
# specify forcings
bg_par_forcing_name="pyyyyy.RpCO2_Rp13CO2"
```

followed by a line that specifies either $\times 1CO_2$ or $\times 3CO_2$, e.g.

```
bg_par_atm_force_scale_val_3=278.0E-06
```

or

```
bg_par_atm_force_scale_val_3=834.0E-06
```

(followed by a line specifying the carbon isotopic composition of the atmosphere).

Similar to as you have seen before for a fossil fuel CO₂ emissions *forcing*, there is a file:

```
biogem_force_restore_atm_pCO2_sig.dat
```

²⁸But running for a full 10,000 years would enable you to follow not only the initial rapid warming perturbation, but also the long-term recovery and adjustment to a new steady state.

and which has contents as so:

```
-START-OF-DATA-
0.0      1.0
999999.0 1.0
-END-OF-DATA-
```

Referring back to the instructions for changing fossil fuel CO_2 emissions *forcing*, you should be able to modify this (or better, copy-rename a new forcing directory and edit the file `biogem_force_restore_atm_pCO2_sig.dat` in that) to create a prescribed time-dependent change in atmospheric CO_2 .

Note that in the format of `biogem_force_restore_atm_pCO2_sig.dat`, the values in the second column scale the value of the parameter `bg_par_atm_force_scale_val_3` in the *user-config*. Hence, in `biogem_force_restore_atm_pCO2_sig.dat`, setting a value of 2.0 in place of 1.0, will double the applied CO_2 forcing. Exactly as per in the fossil fuel CO_2 emissions exercises, pulses, ramps, etc etc can be constructed to control the rate and shape of the applied change in atmospheric CO_2 .

Either way – assess the important and impact of the rate of CO_2 rise and hence warming. (Equally, you might explore rapid cooling and how that fundamentally differs in impact from warming.)

The Oceans, Post end Cretaceous Impact

8.6 Ecology in fake oceans

We can also consider the question of the causes and consequences of different ecologies in the ocean in a more general way and return to our hypothetical or 'fake' oceans.

For any of your 'fake' worlds' that you generated earlier, you can add carbon and nutrient cycling (if that was not already included), plus a marine ecology. To do this, you first need to create a new *base-config* that includes all the carbon cycling and nutrients needed by **ECOGEM** (and **BIOGEM**), then you need to create/configure a suitable *user-config*.

1. *base-config* – First, you need to enable the **ECOGEM** module.

At the top of the your *base-config* file, change the **ECOGEM** 'flag' to true:

```
ma_flag_ecogem=.TRUE.
```

Next, it is likely that you did not add any biogeochemical tracers when you created your fake world, and the *base-config* section:

```
# ****
# TRACER CONFIGURATION
# ****
```

will probably look like:

```
# the total number of tracers includes T and S
# T and S do not need to be explicitly selected and initialized
# ****
# Set number of tracers
GOLDSTEINTRACSOPTS='$(DEFINE)GOLDSTEINTRACS=2',
# list selected biogeochemical tracers
# <<<                                >>>
# list biogeochemical tracer initial values
# <<<                                >>>
```

Go find the *base-config* file muffin.CBE.p0055c.BASES in the configs sub-directory of genie-main. Copy all of the section headed

```
# ****
# TRACER CONFIGURATION
# ****
```

into your *base-config*, replacing the original contents with only 2 tracers selected.

2. *user-config* – Next, you need to define some biogeochemical cycling PLUS and ecosystem.

Go find the *user-config* file: wilsonetal.p0055c.8P8Z.pal.3x in genie-userconfigs sub-directory MS/wilsonetal.2018.

Easiest, is to simply re-use (copy-rename) the *user-config* file: wilsonetal.p0055c.8P8Z.pal.3x. The only parameters you might want to adjust²⁹, other than a different ecological structure (and eg_par_ecogem_plankton_file), is atmospheric CO_2 , which is set to $\times 3CO_2$ by:

```
bg_par_forcing_name="pyyyzz.RpCO2_Rp13CO2"
bg_par_atm_force_scale_val_3=834.0E-06
bg_par_atm_force_scale_val_4=-4.9
```

(and atmospheric $\delta^{13}C$ to -4.9).

Strictly, the scaling for the air-sea gas exchange coefficient, for a fake world, should be:

```
# re-scale gas transfer coefficient ...
bg_par_gastransfer_a=0.722
```

(changing the value from 0.5196 to 0.722).

²⁹Note that ocean alkalinity is also set for an Eocene world.

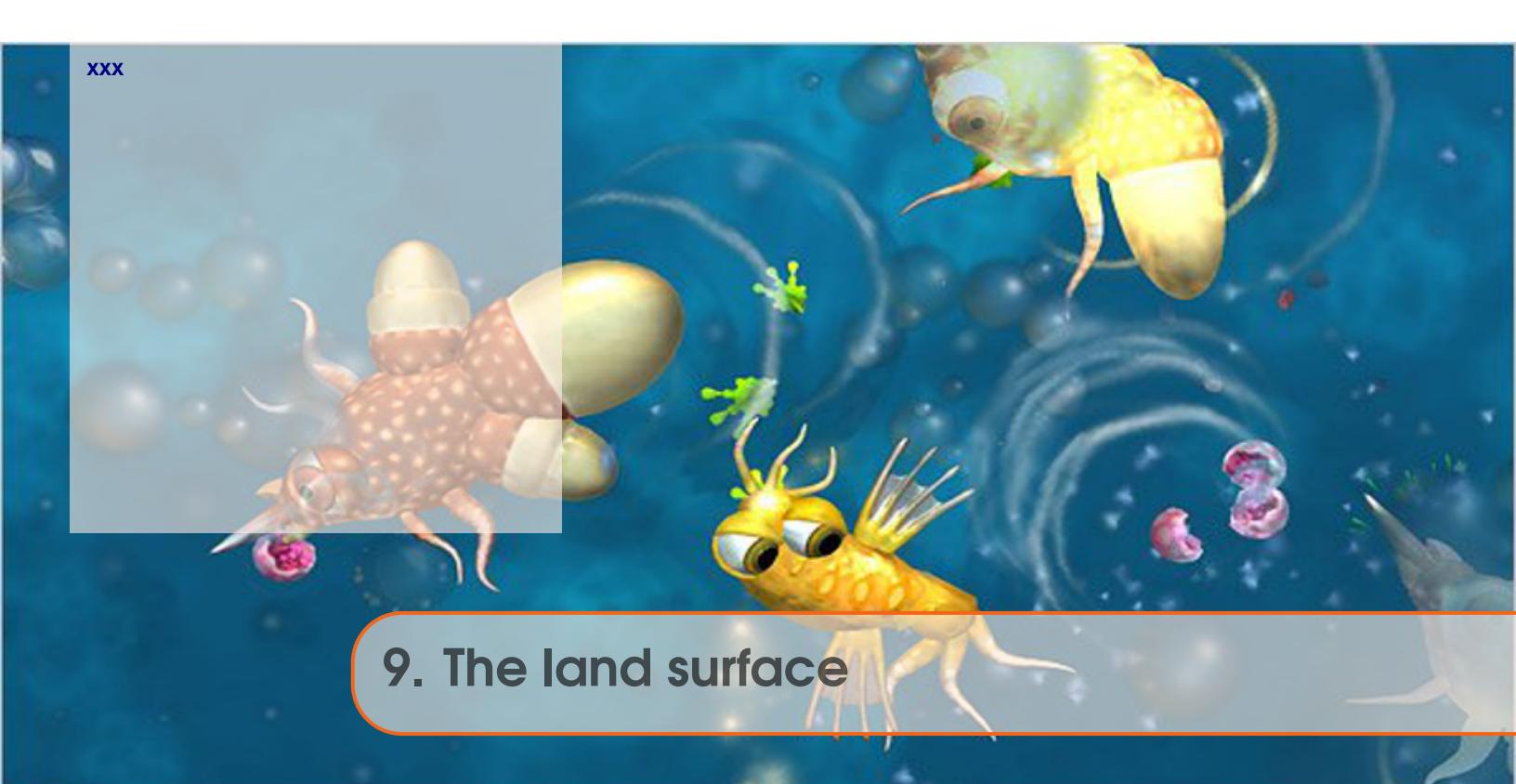
Those changes – enabling **ECOGEM** and adding ocean (and atmosphere) tracers to your *base-config*, and then taking a paleo **ECOGEM user-config** as a template to work from, should get you going with an ecology in your fake world.

If you also want to diagnose ocean circulation better and add a ventilation tracer, then in the *base-config*, increase the number of selected tracers by 2 (under # Set number of tracers) and add the following 2 lines to the list of selected tracers:

```
gm_ocn_select_48=.true. # r  
gm_ocn_select_49=.true. # b
```

and ... in the *user-config*, add (anywhere, but e.g, in the MISC parameter section):

```
# add ventilation tracers  
bg_ctrl_force_ocn_age=.true.
```

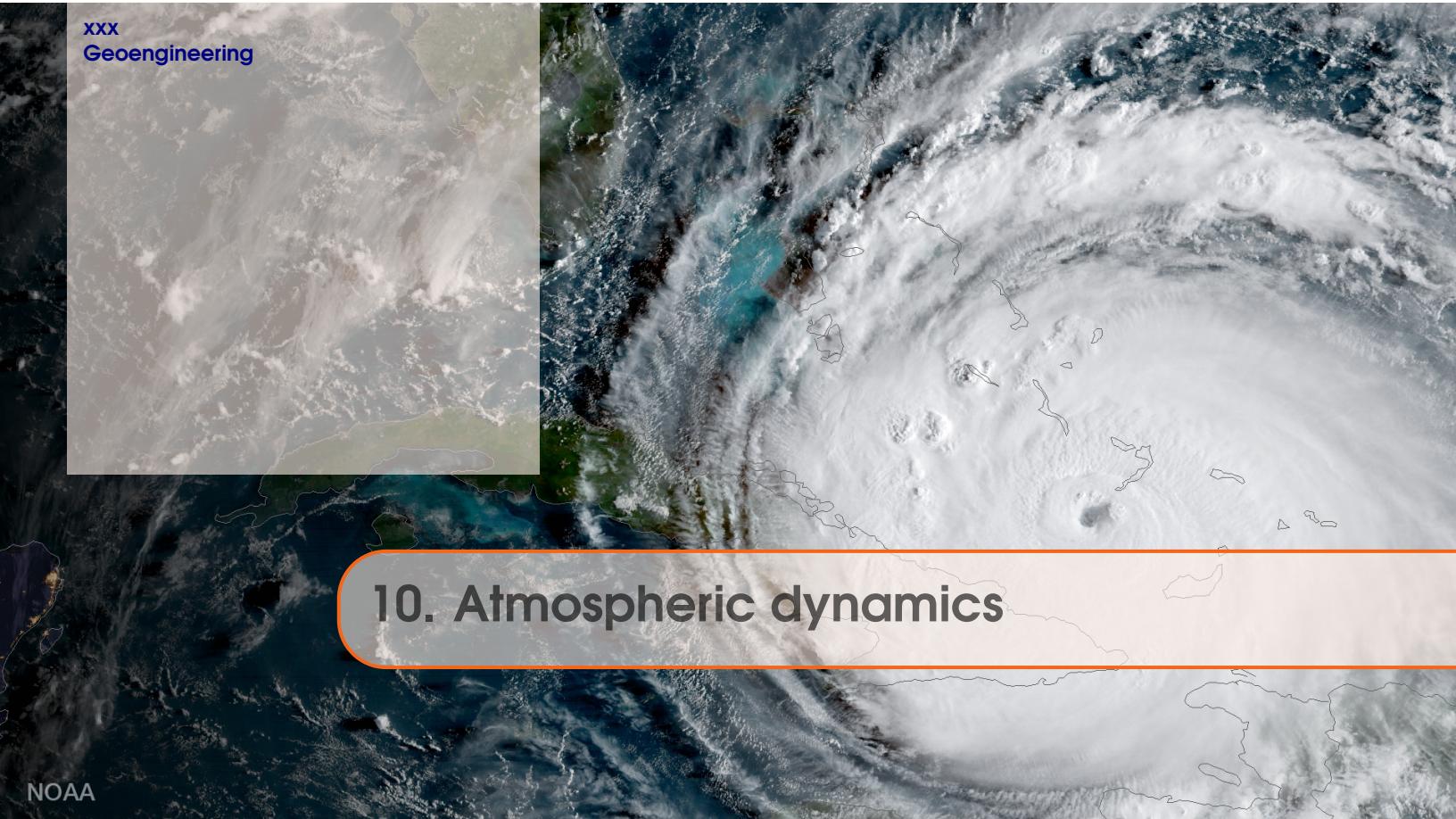


9. The land surface

9.1 xxx

123

9.1 xxx



10. Atmospheric dynamics

NOAA

10.1 **xxx**

10.2 Geoengineering

The long tail of CO_2 (and other tales from the sediments)

Sediment model output

Quantifying how long is the long tail of CO_2

Sediments of the modern Earth

The marine geology of fake worlds

Further ideas

sea ice

open ocean

coastal seas

marine biota

ocean

sediments

terrestrial biota

soils

land surface and hydrology

ice sheet

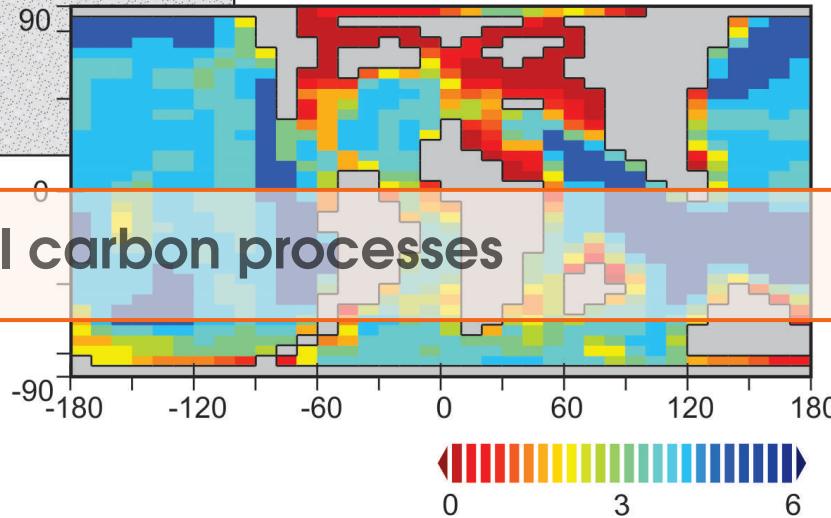
partially-filled upper-most layer

bioturbated zone of 1 cm sediment stack layers

bioturbated zone of buried layers

bioturbation mixing

11. Geological carbon processes



README

You will need to download a new *re-start* file. To fetch this – change to the cgenie_output directory, and type:

```
$ wget http://www.seao2.info/cgenie/labs/Yale.2016/EXAMPLE.\_rwlma.P04\_S18x18.SPIN2.tar.gz
```

Extract the contents of this archive by typing:

```
$ tar xfzv EXAMPLE.\_rwlma.P04\_S18x18.SPIN2.tar.gz
```

You'll then need to change directory back to genie-main directory in order to run **muffin**.

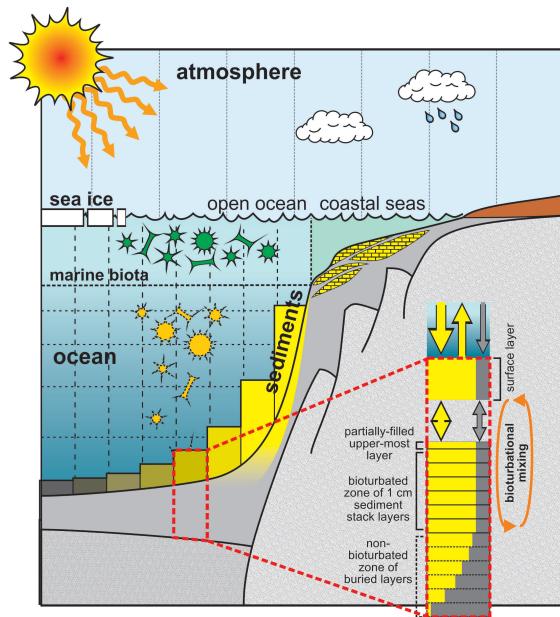


Figure 11.1: Schematic of **SEDGEM** sediment component.

11.1 The long tail of CO_2 (and other tales from the sediments)

This chapter introduces the marine sediment model component in **muffin** – **SEDGEM** (SEDiment GEochemistry Model) plus **ROKGEM**, the terrestrial weathering module. The model experiments now include, a representation of deep-sea sediments and interaction between the preservation and burial of $CaCO_3$ and ocean chemistry, plus balance between weathering and sedimentary burial. For an over-view of the sediment model and what time-scales and nature of carbon cycle interaction between ocean and sediment you can expect, read: *Ridgwell and Zeebe [2005]* and *Ridgwell and Hargreaves [2007]*. **ROKGEM** is described in *Colbourn et al. [2013]*.

Take the new model for a test drive by running on from the provided *re-start* experiment:

EXAMPLE._rwlma.PO4_S18x18.SPIN2.tar.gz. This is a steady-state climate+carbon cycle experiment that includes the deposition of $CaCO_3$ in deep-sea sediments and the balance between weathering (solute input to the ocean) and burial (output). Try running ('briefly', but 100 years would not be too tedious for this faster configuration!):

```
$ ./runmuffin.sh cgenie.eb_go_gs_ac_bg_sg_rg_gl._rwlma.BASES
LABS LAB_11.EXAMPLE 100 EXAMPLE._rwlma.PO4_S18x18.SPIN2
```

Note that the *base-config* file *cgenie.eb_go_gs_ac_bg_sg_rg_gl._rwlma.BASE* specifies the use of a sediment model – the ‘*sg*’ in the filename. Also note that we are running at a degraded 18×18 resolution, requiring fewer (twice as few) time-steps per year – helpful in being able to run **muffin** on geological time-scales but within a reasonable about of real time. (The configuration also utilizes a conceptual/idealized continental configuration somewhat similar to as in the snowball Earth experiments.)

The *user-config* *LAB_11.EXAMPLE* is set up with the global carbon cycle configured as ‘open’ – that is to say, that there is an input of carbon (and alkalinity) to the ocean from weathering, and

a loss due to preservation and burial of $CaCO_3$ in deep-sea sediments. Depending on the state of ocean chemistry (and biology) and weathering, these two fluxes (input vs. output) do not have to balance, and hence ocean carbonate chemistry (and in turn, atmospheric pCO_2) can evolve with time. The *spin-up* may not necessarily have the two fluxes perfectly balanced and hence before you run any experiments you might want to confirm whether the spin-up provided really is adequately 'spun-up'¹.

Note that a residual drift can be dealt with if it is relatively small and near linear and you have a control experiment, because any experiment you carry out will likely also incorporate (or be biased) by the same residual drift. Running a control gives you something to directly contrast your experiment with and calculating the difference (e.g., a difference map or simple subtraction of global numbers) will give you the effect of whatever parameters you changed in the experiment but corrected for any drift. In previous exercises, we were a little lazy, and difference maps were often created with respect to year 1 of an experiment – strictly, they should have been created relative to the same year of a parallel control experiment, i.e., results at year 100 should have been contrasted with the year 100 results of the control.

¹i.e. you might plot some of the variables from the results of the *spin-up* experiment as a function of time and judge whether they are sufficiently converged or not.

11.2**Sediment model output**

There is a whole new set of additional outputs from this configuration of **muffin**, particularly sediment-specific output from the **SEDGEM** module which is saved in the `sedgem` sub-directory of the main experiments results directory. Data saving differs from **BIOGEM** in that the composition of the sediments (and other diagnostics such as rain and dissolution fluxes) is saved only at the very end of a model experiment (hence unlike **BIOGEM**, which saves a series of time-slices throughout the course of a model experiment). So if you kill a run before the very end (or the run crashes), you will get no (or little) **SEDGEM** output.

2D (e.g. surface sediment properties, fluxes, etc.) results can be found in the `sedgem` sub-directory of your experiment directory in a netCDF file called `fields_sedgem_2d.nc`. (Note that there is some duplication of results saving, because a series of *time-slices* of sediment composition are also saved in the 2D **BIOGEM** netCDF file `fields_biogem_2d.nc`. **BIOGEM** also saves a selection of *time-series* of sediment properties – `.res` files starting `biogem_series_sed`. For example, the *time-series* file: `biogem_series_sed_CaCO3.res` contains information about how the mean $CaCO_3$ content of surface sediments evolves with time.

The 2D distribution of *wt% $CaCO_3$* – which is the weight fraction of calcium carbonate ($CaCO_3$) in the surface sediments of the deep ocean (i.e., how much plankton carbonate shell material is there compared to other stuff in the mud at the bottom of the ocean) is saved under a variable called: `sed_CaCO3`. How much carbonate material there is tells you both something about how many carbonate shell secreting plankton were living at the ocean surface above and what is the chemistry of the deep ocean like that these tiny shells were preserved and did not dissolve.

The model also generates artificial sediment ‘cores’ (e.g. see: *Ridgwell* [2007]) and hence what one might expect to see of your applied perturbations recorded in a sediment core recovered from the ocean floor. In the **SEDGEM** results sub-directory, there is a netCDF file which contains all the locations selected (if any) — `sedcore.nc`. These are not really aligned with latitude as the **Panoply** display might suggest – the locations are in fact distributed from all over the ocean (**Panoply** is being fooled in order to display them together). In the **SEDGEM** 2D netCDF file, these locations are marked in the netCDF variable `grid_mask_sedcore`. The locations of these cores are stored in a file containing a little ASCII ‘map’ of the ocean.

- If you are using a ‘paleo’ configuration of **muffin**, indicated by a parameter section in the `base-config` file headed by something looking like:

```
# ****
# GRID & BOUNDARY CONDITION CONFIGURATION
# ****
# insert the automatically generated muffingen parameter list here
# ****
##### cGENIE .config file parameter lines generated by muffingen v0.84 on: 191119 #####
then there will be a parameter line that direct muffin to look for SEDGEM configuration files
in the respective genie-paleo directory, e.g.:
sg_par_pindir_name='../../cgenie.muffin/genie-paleo/fkh_pp03/'
```

- Otherwise, the file lives in: `cgenie.muffin/genie-sedgem/data/input`

The filename is given by the parameter: `sg_par_sedcore_save_mask_name`

Simply be editing (using the ASCII text editor) a ‘0.0’ to a ‘1.0’, you can get the model to generate and save a sediment ‘core’ at that particular (i,j) location.

sedcore.nc variables include:

- phys_layer — sediment layer number (counting down).
- phys_depth -- (cumulative) depth below surface, measured from the sediment surface to the mid-point of each sediment layer (cm).
- th (cm) – thickness of each sediment layer (cm).
- age_CaCO₃ – the mean age of CaCO₃ particles in a sediment layer.
Note that this will not be defined if there is no CaCO₃ preserved.
- ... then some alternative ways of assigning a chronology to a sediment core ... (ignore) ...
- phys_porosity – sediment layer density (as if you cared!).
- sed_POC and sed_POC_13C – mean organic matter content of each sediment layer and its δ¹³C. But note: in this configuration no organic matter is preserved (hence all zeros for POC).
- sed_CaCO₃ and sed_CaCO₃_13C – mean CaCO₃ content (wt%) of each sediment layer and its δ¹³C.
- sed_det and sed_ash – the wt% detrital and ‘ash’ contents of a layer (ash is used as a conservative numerical sediment tracer in order to mark the depth of the start of the experiment).

Obviously – you could plot e.g. CaCO₃ (or its δ¹³C) as a function of depth and/or age across and see how your carbon release experiment might be recorded in the marine geological record (e.g. how does this compare with observations across the PETM?).

Note that the sediment cores reflect not only the material which has accumulated (or not, if it has dissolved ...) during the course of your experiment, but also the material that accumulated during the 50,000 year spin-up. AND, whatever material the sediment core was initialized with to start with. For example, the large interval of initially 100% detrital material at the base of the sediment core simply reflects the initialization of the sediment array in the model. Also note the ash ‘peak’ near the bottom of the stack (filled) sediment layers – this is a tracer to ‘tag’ the start of the model spin-up. If you look at the spin-up results (not your recent perturbation experiment) – the ash peak lies in a sediment layer with age 50,000 years. But why is there any ash deeper than the age corresponding to the start of the spin-up? How can it get there?

11.3 Quantifying how long is the long tail of CO_2

You are now considering a rather more complex carbon cycle than before (e.g. now including a number of additional, mostly sediments/weathering processes). It is hence worth conducting some number of idealized perturbations of the global carbon cycle to get a feel for the sensitivity and time-scale of the system response.

For instance – one illustrative experiment, and which has a parallel to experiments you have conducted previously, is to add a pulse CO_2 release to the atmosphere and track the consequences for atmospheric pCO_2 and ocean chemistry (particularly ‘alkalinity’), and now also e.g. deep sea sediments. To the LAB_11.EXAMPLE *user-config*, add the lines:

```
bg_par_forcing_name='pyyyzz.FpCO2_Fp13CO2'
bg_par_atm_force_scale_val_3=8.3333e+016
bg_par_atm_force_scale_val_4=-27.0
```

in order to achieve a 1000 PgC total release in a single year (or multiply by 5 to be comparable-ish to the 5000 PgC release in *Ridgwell and Hargreaves* [2007]) with an isotopic composition of -27‰ – appropriate for a fossil fuel carbon source. For reference – methane derived carbon (e.g. as from hydrates) would be more like -60‰ .

BE CAREFUL – if you have previously edited files of the *forcing* pyyyzz.FpCO2_Fp13CO2 , you are going to need to return whatever you have edited back to its original state. Or, carry out the following procedure:

1. From genie-main, \$ make cleanall
2. Then head up a directory level (cd ..) to cgenie.muffin
3. Type: \$ git status -uno

You should see a list of all files that you originally installed (cloned from GitHub) that you have changed. e.g. you might see:

```
modified:   genie-forcings/pyyyzz.FpCO2_Fp13CO2/biogem_force_flux_atm_pCO2_sig.dat
```

(Note that file you create are not listed. TO see those, type: \$ git status)

4. To restore the state of any file that you might have edited, git checkout that file out, e.g.:

```
$ git checkout genie-forcings/pyyyzz.FpCO2_Fp13CO2/biogem_force_flux_atm_pCO2_sig.dat
```

Run the model for as long as you dare (or can be bothered) – 1,000 or 2,000 years might be just enough as a minimum to start to see impacts on deep-sea sediments, but 5,000 or 10,000 years would be much better. (You can always submit this to the cluster queue and get on with something else.) FYI: 10,000 years is going to take something like an hour ... if you are lucky ...

Plot the time-series of e.g. atmospheric pCO_2 and compare to the (much shorter experiments) you have carried out before with a simple ocean+atmosphere only system. Compare how quickly atmospheric pCO_2 decays compared to previously **muffin** publications² (e.g. *Ridgwell and Hargreaves* [2007]) or other models (e.g. *Archer et al.* [2009]) and how the sediments respond (e.g. the time-series of sediment $CaCO_3$ content).

²see: <http://www.seao2.info/pubs.html>

To properly (quantitatively) appreciate the role of ocean-sediment interaction (and weathering) and controlling atmospheric pCO_2 , you need to contrast these experiments with as similar a model configuration as possible – for instance, one that is identical excepting having no sediments (or weathering). You can achieve this quite simply: create (/copy-rename) a new *user-config* based on LAB_x.sediments and edit the lines³:

```
# set an 'OPEN' system
bg_ctrl_force_sed_closesystem=.false.
```

changing it to:

```
# set a 'CLOSED' system
bg_ctrl_force_sed_closesystem=.true.
```

What this does is to force the model to always maintain an exact balance between the preservation and burial in marine sediments of $CaCO_3$, with the supply of solutes derived from the weathering of $CaCO_3$ on land. Because no excess or deficit of weathering vs. sedimentation is allowed to occur, no changes in ocean chemistry (other than by air-sea gas exchange) occur. This configuration hence acts (geochemically and dynamically) exactly the same way as a configuration without any sediments or weathering being present (and as used previously).

By comparing the two experiments: can you deduce the effect of the sediments in modulating (accelerating or decelerating) atmospheric pCO_2 decline?⁴ Also view the sediment distribution (of $CaCO_3$): what are the impacts on sediment composition in the case of an experiment configured with an ‘open’ system (vs. a ‘closed’ system)? Here, the time-series file of mean global sediment composition biogem_series_sed_CaCO3.res (wt% $CaCO_3$) may help illustrate what is going on here. Note that the way the ‘closed’ system is constructed; a response of the sediments is predicted and saved in the output, even though it is not allowed to affect chemistry or atmospheric pCO_2 .

To recap – you are aiming to run a CO_2 emissions experiment using a **muffin** configuration including variable weathering and sedimentation (LAB_11.EXAMPLE) with a version/copy that you have edited to create a ‘closed’ ocean-atmosphere system, with no ocean-sediment or weather feedbacks on atmospheric CO_2 . Ideally, you would also run a pair of control experiments – both ‘open’ and ‘closed’ configurations but with no CO_2 emissions specified. (A total of 4 experiments.)

³You do not have to edit the comment line (#) but it will help you remember what you have done.

⁴e.g. you could compare the pCO_2 time-series of the 2 different experiments.

11.4 Sediments of the modern Earth

Important is to critically and quantitatively assess to what degree **muffin** provides an adequate representation of the interaction between ocean chemistry and sediment composition (e.g., in CaCO₃ buffering of CO₂ release and 'carbonate compensation'). Key to this, is contrasting model output against observational-based data. Such an approach is presented in *Ridgwell and Hargreaves [2007]*.

The required *base-* and *user-config* files are provided as part of the **muffin** code release:

- cgenie.eb_go_gs_ac_bg_sg_rg.worbe2.BASE – the *base-config* file, including **SEDGEM** and **ROKGEM** modules.
- EXAMPLE.worbe2.RidgwellHargreaves1997_S36x36.SPIN1 – the *user-config* for the 1st stage (20 kyr long) *spin-up* described in *Ridgwell and Hargreaves [2007]*. This file lives in the *genie-userconfigs* sub-directory.
EXAMPLE.worbe2.RidgwellHargreaves1997_S36x36.SPIN2 is the 2nd-stage, 50 kyr *spin-up* that uses EXAMPLE.worbe2.RidgwellHargreaves1997_S36x36.SPIN1 as a *re-start*.

e.g. you would run the 2 experiments:

```
./runmuffin.sh cgenie.eb_go_gs_ac_bg_sg_rg.worbe2.BASE /  
    EXAMPLE.worbe2.RidgwellHargreaves1997_S36x36.SPIN1 20000  
  
./runmuffin.sh cgenie.eb_go_gs_ac_bg_sg_rg.worbe2.BASE /  
    EXAMPLE.worbe2.RidgwellHargreaves1997_S36x36.SPIN2 50000  
    EXAMPLE.worbe2.RidgwellHargreaves1997_S36x36.SPIN1
```

Does it get the broad patterns right (is it more right than wrong, or more wrong than right)? Do you think the model-data misfits might be important?

11.5 The marine geology of fake worlds

You can configure any of your previous fake worlds, or generate new ones, to have a full geologic carbon cycle including deposition of $CaCO_3$ in marine sediments and weathering on land.

In order to generate the requisite **SEDGEM** configuration files, you need the following settings in your **muffingen** configuration file:

- `opt_makeseds=true`; – requests that **SEDGEM** files are generated.
- `par_sedsoft=2`; – requests that a randomized bathymetry is generated. This is needed because fake worlds have by default, a flat bathymetry.

Instead, if you 'draw' a non-uniform bathymetry, you would set:

`par_sedsoft=0`; – requests that the ocean depth levels ('`k1`' file') are used to inform the ocean floor depth assumed by **SEDGEM**.

(Option 1 is most commonly used in conjunction with a GCM-derived continental configuration.)

So, if you define continents in your fake world, but do not change the bathymetry in **muffingen**, `par_sedsoft` option 2 will give you a randomized distribution of sediment depths (used in the $CaCO_3$ solubility pressure calculation only). Selecting option 0 will simply translate your chosen **muffingen** ocean depths into **SEDGEM** depths. Note that you can still have a flat bottom to the ocean (no variation in ocean floor depth) and choose option 0.

If you define a specific bathymetry by hand (e.g. draw in ocean ridges), you probably do not want it over-written (by a random **SEDGEM** depth pattern) and hence you should chose option 0.

For speed, it is recommended that you generate your **muffingen muffin** configurations at the lowest reasonable resolution – $18 \times 18 \times 8$ would be suitable – 18×18 resolution in lon vs. lat, and 8 levels in the ocean. (You could try pushing this a little further and more extreme, e.g. $12 \times 12 \times 8$.) Note that when you create your *base-config*, you should use the template:

`CONFIG_template_08lvl_R07.config`

(for the 8-level ocean, rather than `CONFIG_template_16lvl_R07.config` which is designed for 16-ocean level configurations).

Once you have copy-pasted the configuration output of **muffingen** into the template *base-config* file (and suitably renamed it), you need to enable the geologic carbon cycle modules (**SEDGEM** and **ROCKGEM**). At the top of the *base-config* file, ensure that the following are set:

```
ma_flag_sedgem=.TRUE.  
ma_flag_rokgem=.TRUE.
```

And then, further down and under the heading:

```
# TRACER CONFIGURATION
```

you need to define how many 'tracers' in the ocean, what they are, and any initial values that differ from modern defaults. Plus, whatever atmospheric and sediment tracers you want (plus default values in the atmosphere). Simplest at this point is to copy-paste from an existing *base-config*, such as:

```
cgenie.eb_go_gs_ac_bg_sg_rg.worbe2.BASE
```

(which is the *base-config* used in Ridgwell and Hargreaves [2007]).

As for a suitable *user-config* ...

... you could take the *user-config* from *Colbourn et al.* [2013] (itself a modification of Ridgwell and Hargreaves [2007]) – available as part of the **muffin** code release, as file:

EXAMPLE.worbe2.Colbournetal2013.CTRL (genie-userconfigs sub-directory)

Then generalize this according to the paleo sediment configuration used in *Ridgwell and Schmidt* [2010] (file: EXAMPLE.p0055c.RidgwellSchmidt2010.SPIN1 in the genie-userconfigs sub-directory) (i.e. making CaCO_3/POC export invariant, adding a fixed detrital flux of the sediments). Cutting the lines down to the bare minimum (excepting comments pertaining to new/altered lines), a suitable *user-config* then looks like:

```
# --- CLIMATE -----
# enable CO2 climate feedback
ea_36=y
# --- BIOLOGICAL NEW PRODUCTION -----
bg_par_bio_k0_P04=1.9582242E-06
bg_par_bio_c0_P04=2.1989611E-07
# --- ORGANIC MATTER EXPORT RATIOS -----
bg_par_bio_red_DOMfrac=0.66
# --- INORGANIC MATTER EXPORT RATIOS -----
# set fixed export  $\text{CaCO}_3$  as a proportion of particulate organic matter
bg_par_bio_red_POC_CaCO3=0.200
bg_par_bio_red_POC_CaCO3_pP=0.0
# --- REMINERALIZATION -----
bg_par_bio_remin_DOMlifetime=0.5
bg_par_bio_remin_POC_frac2=6.4591110E-02
bg_par_bio_remin_POC_eL1=550.5195
bg_par_bio_remin_POC_eL2=1000000.0
bg_par_bio_remin_CaCO3_frac2=0.468
bg_par_bio_remin_CaCO3_eL1=1083.486
bg_par_bio_remin_CaCO3_eL2=1000000.0
# --- SEDIMENTS -----
# sediment diagenesis options
sg_par_sed_diagen_CaCO3opt="ridgwell2001lookup"
sg_ctrl_sed_bioturb=.true.
sg_ctrl_sed_bioturb_Archer=.false.
sg_par_n_sed_mix=20
sg_par_sed_mix_k_sur_max=0.15
sg_par_sed_mix_k_sur_min=0.15
# additional detrital flux (g cm-2 kyr-1)
sg_par_sed_fdet=0.180
# --- WEATHERING -----
# set a 'OPEN' system
bg_ctrl_force_sed_closedsystem=.false.
# set  $\text{CaCO}_3$ _weathering-temperature feedback
rg_opt_weather_T_Ca=.TRUE.
# set  $\text{CaSiO}_3$ _weathering-temperature feedback
rg_opt_weather_T_Si=.TRUE.
# weathering reference mean global land surface temperature (C)
rg_par_ref_T0=8.48
#CO2 outgassing rate (mol C yr-1)
rg_par_outgas_CO2=5.59E+12
# global silicate weathering rate (mol Ca2+ yr-1)
rg_par_weather_CaSiO3=5.59E+12
# global carbonate weathering rate (mol Ca2+ yr-1)
rg_par_weather_CaCO3=5.59E+12
# d13C
rg_par_outgas_CO2_d13C=-6.0
rg_par_weather_CaCO3_d13C=12.8
# --- DATA SAVING -----
bg_par_data_save_level=4
bg_ctrl_debug_lv10=.true.
ma_debug_loop=1
# --- END -----
```

Here, the reference temperature for weathering is set for a modern-like continental configuration:

```
# weathering reference mean global land surface temperature (C)
rg_par_ref_T0=8.48
```

If left un-changed, then the negative silicate weathering feedback will operate on atmospheric CO_2 – letting it accumulate, or removing it, until the reference temperature is achieved. If instead, you want to achieve a specific atmospheric CO_2 at steady state, you need to know the relevant mean global land surface temperature. To determine this – first run **muffin** with a prescribed atmospheric CO_2 value, e.g. by adding:

```
# --- FORCINGS -----
# specify forcings
bg_par_forcing_name="pyyyyz.RpCO2_Rp13CO2"
bg_par_atm_force_scale_val_3=278.0E-06
bg_par_atm_force_scale_val_4=-6.5
```

(or some other value of CO_2) to your *user-config*. Running the model for about 1000 years should be more than enough to achieve a near steady-state of surface climate, allowing to read off the mean global surface air temperature over land (BIOGEM file *biogem_series_misc_SLT.res*), and set the reference temperature to this value. You can then run an experiment without a prescribed atmospheric CO_2 forcing, now knowing that the silicate weathering feedback will always act to restore atmospheric CO_2 back to that value.

Despite the low grid resolution, this is still going to take a long time for particularly much to happen and you could probably do with some (numerical) 'help'. You can accelerate the run-time of **muffin** in calculating the balance between weathering and sedimentation, a-la *Lord et al.* [2015]. To do this, you need to add a new section of parameter choices to the *user-config* file:

```
# --- GEOCHEM ACCELERATION -----
gl_ctrl_update_pCO2=.true.
ma_gem_notyr_min=10
ma_gem_notyr_max=10
ma_gem_yr_min=90
ma_gem_yr_max=90
ma_gem_dyr=0
ma_gem_adapt_auto=.false.
```

Here, this specifies that you would like to spend only 10 years of full updating of the model for each 90 years spent in an accelerated calculation which treats the entire ocean as if it were a single geochemical reservoir (see: *Lord et al.* [2015]) – an almost 10 : 1 acceleration and reduction in experiment run-time. (But note that the ocean circulation and ocean carbon and nutrient cycles will appear to be taking longer to come to equilibrium because they are only being fully updated 10 years in every 100).

Note that this acceleration is good for spin-ups and determining equilibrium weathering-sedimentation states, but not so useful for transient CO_2 changes (see in *Lord et al.* [2015] for how a less acceleration and adaptive setup had to be used). Similarly, acceleration is not so useful for varying orbits experiments.

A couple of different possible experimental investigations follow.⁵

1. Firstly, you could simply test what happens if there is no sediment surface depth variability as compared to a configuration with depth (pressure) variability in sediment surface depth (and one that follows a modern-like distribution of depth vs. area). The science question would be something like – how important is a distribution of ocean depths to setting the steady state alkalinity (or saturation) of the ocean? Or: in having an appreciable amount of shallower sea-floor where $CaCO_3$ will be preserved and buried much more readily – how much less saturated must the ocean as a whole become in order to balance weathering and sedimentation? For this, you'll need 2 **muffingen** configurations, each with the same continental configuration, but one generated with random sediment depths, and one with a uniform depth.

Simplest, but more tedious, is to run **muffingen** twice, with your configuration settings files differing only in the value of `par_sedsopt`. However, faster is to generate one configuration using the configuration template `EXAMPLE_BLANK.m`, and then generate a 2nd one, using the '`k1`' file generated by the first **muffingen muffin** generation. An example of creating a **muffin** configuration from a '`k1`' file is: `EXAMPLE_K1_permian.m`. Basically, the only thing that changes between the first and second **muffingen muffin** generation, in addition to `par_sedsopt`, is the value of `par_gcm`, which changes from '' (empty) in `EXAMPLE_BLANK.m`, to '`k1`' in `EXAMPLE_K1_permian.m`.

2. As a variant on the above – you might 'draw' a large and relatively shallow (e.g. 1 or 2 km depth) sea-floor plateau, run the model, and determine how much this has influenced ocean chemistry, at steady state (remembering to run an experiment with no sea-floor plateau in a second experiment as a control).
3. How important is continental configuration and the position of the continents in controlling atmospheric CO_2 via weathering?

You might, for instance, create 2 different continental configurations in **muffingen**, both with the same cratonic (land surface) area, and explore whether you achieve a different steady state atmospheric CO_2 depending on whether the continent is at the pole, or centered on the equator. Remember that the parameterized silicate weathering feedback acts to restore global mean surface air temperature over land, to the reference value.⁶ You might then guess the sign of the change in atmospheric CO_2 given that the weathering will act to restore the polar and equatorial continental temperatures to the same value and it does that by causing atmospheric CO_2 to change. (But perhaps the magnitude of the effect will be unexpected, which is why you need a model.)

⁵Note that in using the given *user-config* settings, steady state is only reached after several 100 kyr (which is going to take a while, even at low resolution and with acceleration). Regardless of your assumptions regarding sea-floor bathymetry, you should find that atmospheric CO_2 always reaches the same value.

⁶Note that the simple weathering parameterization in **muffin** does not care how much land there is, only what the mean surface air temperature over that land is.

11.6 Further ideas

You might further explore the role of weathering and sensitivity the sensitivity of atmospheric pCO_2 and climate to the strength of the weathering feedback as well as to the assumed rate of volcanic CO_2 outgassing, as well as how the products of weathering on land are accommodated through the burial (and pattern of burial) of $CaCO_3$ in deep-sea sediments. Parameters of ‘interest’ here, i.e. ones that you might adjust to explore the silicate weathering feedback and long-term controls on atmospheric pCO_2 , are listed under # -- WEATHERING -- in the *user-config*, and include:

- rg_par_outgas_CO2 -- the global CO_2 outgassing rate in units of $molyr^{-1}$.
- rg_par_ref_T0 -- the reference land surface temperature for weathering (units of $^{\circ}C$).

In **muffin**, global weathering rates will increase or decrease depending on whether the mean global surface temperature (which is reported and saved in **BIOGEM time-series** output file: bio-gem_series_misc_SLT.res). So, if you increase the reference temperature, weathering rates will drop and CO_2 will accumulate in the atmosphere until the Earth was warmed sufficient that the mean global land surface temperature again matches the reference temperature. And *vice versa* for the case of reducing the reference temperature. Maybe try a pair of experiments (plus a control in which you do not adjust the reference temperature) in which you adjust the value of rg_par_ref_T0 both up and down, to confirm this.

Conversely, increasing or decreasing the rate of CO_2 outgassing should also act to increase or decrease, respectively, global temperatures (at steady state). Another pair of experiments (plus a 3rd, control) would be to try increasing and decreasing the value of rg_par_outgas_CO2.

Another pair of important controls on the preservation and burial of $CaCO_3$ in marine sediments (but not one that ultimately at steady state, affects CO_2 and climate), are the fluxes of $CaCO_3$ and organic carbon (POC) to the sediment surface.

All other things being equal – increasing the export of $CaCO_3$ and hence flux to the sediment surface requires that fraction of $CaCO_3$ that dissolves increases at steady state. In other words – instantaneously increasing the flux to the sediments of $CaCO_3$ will increase burial relative to the weathering flux. The excess sink of $CaCO_3$ will act to lower DIC and ALK (and Ca^{2+}), lowering the carbonate saturation state of the ocean, and reducing the preservation of $CaCO_3$. This will continue up to the point where the preservation and burial of $CaCO_3$ once again balances weathering.

An interesting question is what happens to atmospheric CO_2 . Ultimately it must return to its initial value due to the silicate weathering feedback. However, this occurs on a long (ca. 200 kyr) time-scale, whereas an imbalance between $CaCO_3$ weathering and burial can act to change ocean chemistry on much shorter (1-10 kyr) time-scales. To explore this, you can adjust the parameter that sets the ratio of $CaCO_3$ to POC export – having the effect of changing the $CaCO_3$ flux to the sediments whilst keeping everything else (almost) in the ocean carbon cycle invariant. The parameter is:

`bg_par_bio_red_POC_CaCO3=0.200`

Adjusting this value higher will increase the global export of $CaCO_3$ from the surface ocean.

It is much more difficult to adjust the POC flux independent of $CaCO_3$ (i.e. keeping the global $CaCO_3$ flux invariant) because of the way **muffin** parameterizes biological $CaCO_3$ export as a function of POC export. Without perturbing the ocean PO_4 and hence productivity and carbon cycling too much, you can shift slightly more or less POC to a form that is assumed to reach the sediment surface without degradation – i.e. increasing this fraction results in a higher proportion of exported POC reaching the sediment surface, and reducing the fraction decreases the POC flux to

the sediments and hence organic matter available to help drive $CaCO_3$ dissolution. This parameter is:

```
bg_par_bio_remin_POC_frac2=6.4591110E-02
```

Here, by default, just 0.065, or 6.5% of organic matter exported from the surface ocean always reaches the sediment surface unaltered. (In addition, and particularly at shallowed ocean floor depths, some of the other 93.5% can also reach the sediment surface. See *Ridgwell et al.* [2007].)

You might ... investigate other facets of the nature of the relationship between ocean and sediments (and weathering) as how climatic (biogeochemical) signals are encoded in the marine geological record. For instance, you could explore the effect/importance of sediment ‘bioturbation’ (e.g. see *Ridgwell* [2007]). Whether the surface sediment layers are bioturbation or not is set by the parameter:

```
sg_ctrl_sed_bioturb=.true.
```

Simply change to `.false.` in order to ‘turn off’ bioturbation. What happens if you then run a CO_2 release experiment? How is the sediment signal different?

Rather than driving an initial dissolution of $CaCO_3$ in deep sea sediments, the opposite response – increased rather than decreased $CaCO_3$ preservation – can be obtained by removing CO_2 from the atmosphere. This can be implemented by a negative rather than positive emissions scaling in the *user-config* (or in the *forcing* itself). BE CAREFUL here, as for a pre-industrial atmosphere with 278 ppm CO_2 , you do not have a lot more than $\sim 600PgC$ in there (the atmosphere) to begin with. So either: remove less than $600PgC$, or remove the carbon over rather little longer than 1 year.

Again – view the time-series of ocean composition (e.g. DIC, ALK, $\delta^{13}C$) as a function of time, plus mean sediment surface composition ($wt\% CaCO_3$). Also view the sediment ‘cores’ and hence what in practice has been incorporated into accumulating sediments as a record of what is a very sharp perturbation at the ocean surface (and atmosphere).

How is an event characterized by CO_2 removal from the system recorded differently from one characterized by CO_2 release? Are there different implications for constructing core age-scales and chronology, e.g. where in (core) ‘time’ does the excursion maximum appear to lie? Do all sediment locations show identical responses (i.e. does it matter what the initial $wt\% CaCO_3$ is?).

12. Proxies and Reconstructing the Past

README

How low can you go?

Global weathering rate.
Iron fertilization.
Remineralization depth.
Macro nutrient inventory and uptake.
CaCO₃:POC rain ratio.
Sea-ice extent.
Atlantic circulation.
Global ocean circulation / 'brine rejection'.
Salinity
Terrestrial carbon storage

13. Synthesis: Earth system dynamics

Stuff to keep in mind:

"There are known knowns.

These are things we know that we know.

There are known unknowns.

That is to say, there are things that we know we don't know.

But there are also unknown unknowns.

There are things we don't know we don't know."

Donald Rumsfeld, former US Secretary of Defense

13.1 README

blah

13.2 How low can you go?

Your task is to explain low glacial atmospheric CO₂.

It is up to you whether you aim for 190 ppm, or want to retain as much consistency with other (paleoceanographic) constraints as possible.

In addition to the control and spin-up user-configs, you have been provided with: exp12_glacial which can be used as a template user-config file for you to base (if you want!) your glacial CO₂ investigations on (at least initially).

Before embarking upon some glacial experiments (below) – other model outputs/fields you might also keep in mind and for which some data/observational constraints on your glacial CO₂ 'solution' may be available (e.g. see Kohfeld and Ridgwell [2009]), include:

- The 2D fields_biogem_2d.nc field: phys_seaice ('sea-ice cover'), for which some glacial sea-ice limit information exists. (CLIMAP)
- The 2D fields_biogem_2d.nc field: ocn_sur_temp ('surface-water temp') (or view the surface ocean layer in the 3D file), for which 2 (one older, one newer) comprehensive datasets exists – it would be reasonable to question whether you achieve an adequate glacial (surface) climate state and if not, whether this impacts any bias (and in which direction) to your CO₂ solution. (CLIMAP, MARGO)
- The 2D fields_biogem_2d.nc field: sed_CaCO3 ('sediment core-top CaCO₃') (also available from the sedgem model output), for which some glacial CaCO₃ distribution data/estimates exist. (Old Catubig paper in GBC)
- The 2D fields_biogem_2d.nc field: phys_opsia (' Atlantic streamfunction'). While poorly resolved in this model configuration, many (glacial) model studies report the circulation field and hence they provide a point of comparison for your GENIE-based research.
- The 2D fields_biogem_2d.nc field: ocn_D DIC_13C (' planktic-benthic difference DIC_13C') and also the individual planktic (surface) and benthic (bottom) fields. A significant amount of d13C data exists in the literature for both glacial and interglacial states.
- The 2D fields_biogem_2d.nc field: ocn_ben_O2 (' bottom-water O₂') (and also horizontal slices in the 3D file). Ideally, no-where in the ocean should anoxia (no oxygen) occur. It certainly should not be widespread across one or more ocean basins if your glacial CO₂ solution is to get published in Nature ;)
- The 2D fields_biogem_2d.nc field: ocn_ben_sal ('bottom-water sal') (and also in the 3D file as horizontal slices), for which some estimates exists for a few places in the ocean. (Jess Atkins Science paper)
- The 2D fields for surface and deep PO₄ (and also in the 3D file as horizontal slices) as some proxy evidence exists for changes in nutrient utilization. (Cd/Ca proxies – e.g. papers by Elderfield and Rickaby)

Also refer to the 3D netCDF files and the time-series where helpful.

Note here that the spin-up provided is 'modern' and hence glacial data cannot be directly contrasted – the above suggestions/guidance are intended as a starting point only.

In your glacial CO₂ investigations, 2 separate initial modifications of the model will nudge it in the vague direction of a glacial state (e.g., see Kohfeld and Ridgwell [2009]):

- A modification of surface (actually 'planetary') albedo to try and take account of some of

the cooling influences of the large (Northern Hemisphere) ice sheets that were present during the last glacial but which are not calculated or explicitly taken into account in the version of cGENIE you are using.

- A modification of greenhouse gas radiative forcing (as per in the snowball Earth experiments) to take into account the lower CO₂, CH₄, and N₂O concentrations in the atmosphere during the last glacial. While you will be attempting to reproduce ca. 190 ppm atmospheric CO₂ (and hence deduce the reasons for low glacial CO₂), you may not necessarily achieve this, and you have no means of explicitly controlling the other A Hitchhikers Guide to the advanced Black Arts (of Earth system modelling) LAB Session VII 3 greenhouse gases, so you may as well get the radiative forcing and hence the glacial climate state as close as possible before trying to adjust the carbon cycle. But it is up to you whether you prefer to not 'cheat' and have whatever CO₂ cGENIE simulates, directly affect climate (and hence feed back on CO₂).

The glacial boundary conditions are implemented as follows:

- To achieve a pseudo-glacial planetary albedo modification, add the following lines to a user-config file (and see Lab V):

```
# adjusted planetary albedo
ea_albedop_offs=0.200
ea_albedop_amp=0.360
ea_albedop_skew=0.0
ea_albedop_skewp=4
ea_albedop_mod2=-15.000
ea_albedop_mod4=-2.500
ea_albedop_mod6=0.000
```

- For glacial radiative forcing, add the lines:

```
# glacial CO2 radiative forcing
ea_radfor_scl_co2=0.6835
# glacial CH4 radiative forcing
ea_radfor_scl_ch4=0.5
# glacial N2O radiative forcing
ea_radfor_scl_n2o=0.8
```

If you like – you can carry out separate experiments to test the effect of each of these in turn and hence to learn the effect and impact on atmospheric CO₂ of each individually before combining them. Ideally, these lines should ultimately be included in all (glacial) experiments that you require glacial albedo and radiative forcing for.

By all means play around with the albedo and radiative forcing climate boundary conditions, although one might wonder the reasoning behind adjusting radiative forcing below e.g. that appropriate for full glacial conditions. However, the planetary albedo is less certain as implemented in the model. For instance, one might legitimately adjust this to achieve appropriate last glacial sea surface temperatures (SST), or rather: a glacial-interglacial difference in SSTs similar between model and data.

By this point, you should have created a new (mostly) spun-up model state, incorporating: (i) higher planetary albedo, and (ii) lower greenhouse gas forcing. The duration of this new, glacial spin-up needs to be sufficient to bring the system into (a new) equilibrium (of which the slowest

adjusting component will be sediment composition (wtwt% CaCO₃ will to some extent be reflected in continuing changes in pCO₂ (why?)).

Before carrying on, check that everything is 'correct' (or at least: understandable) so-far. In particular: confirm that you have a colder ocean (due to altered albedo and/or greenhouse radiation forcing) ... no, seriously! You never know what might have gone wrong with a simple slip of the keyboard ... Surface ocean temperature also has established proxies for its glacial value and so the model can be contrasted against data.

The file biogem_series_ocn_temp.res is the time-series results file for ocean temperature – the 2nd column is the mean ocean temperature, the 3rd column is mean sea surface temperature (SST), and the 4th mean benthic (deep (> 2 km) ocean floor). How much colder has it become? Is this realistic? Analyze the SST distribution (the surface field of the 3D netCDF time-slice file, or the 'sur_*' variables in the 2D netCDF file) – how does this compare to observations? In the book chapter the data-based difference in SST between the LGM and Holocene is given. However, the map given is for the glacial-interglacial difference, which happily is something you have previously learned to do using Panoply (I hope!).

Also, what is the CO₂ impact of lower SSTs? Note that you may not be able to directly compare your CO₂ prediction with all previous studies (e.g. summarized in the book chapter) because in your model, sea-ice and ocean circulation will also have been affected to some extent by the climate change. (How much have they been affected? There are also some proxies for sea-ice extent as well as ideas and hypotheses about ocean circulation changes.) Many (but not all) previous model studies have simply estimated CO₂ changes due to temperature change with fixed sea-ice and circulation fixed, by prescribing a different ocean surface temperature for the CO₂ solubility calculation. (This is a little beyond the scope of what you are expected to do here, but can be done in GENIE.)

Unless you are *extremely* lucky and already have a value of atmospheric CO₂ that is 90 ppm lower than pre-industrial (ca. 278 ppm) ... (WTF?!) – you may want to test other changes that might have taken place between glacials and interglacials that affected CO₂. Obviously a spot of creating new user-config files will be in order here (perhaps using: exp12_glacial as a template, but it is entirely up to you). Ideally, you would test the impact of each change individually first before combining them, so as to develop a better understanding of the different ways in which CO₂ is controlled (and the associated impacts on other elements of the global carbon cycle and climate) before bunging everything in together.

Before diving straight in – note the number of different modifications of the global carbon cycle that might be considered and tested in the model. What is your methodological strategy going to be? Are you going to add all of the modifications into a single run and hope that you can understand what has happened at the end? (I hope not!) Are you going to run all the modifications individually (how?). Are you going to try combinations to test whether any combine non-linearly? You will want to make use of the cluster queue and submit at least some of the experiments. You will also need to already have a good idea of how long to run them before (hopefully you obtained this knowledge from the idealized perturbation experiments in the previous tutorials).

Some suggestions (i.e., this not an exhaustive list, nor a prescribed one and not everything necessarily has to be done!):

13.2.1 Global weathering rate.

Refer to Ridgwell and Zeebe [2005] for the role of weathering. Also to Kohfeld and Ridgwell [2009] for some references to the changes in weathering that might have taken place between glacial and interglacial. The namelist parameter that controls the annual rate of solute input into the ocean is:

`rg_par_weather_CaCO3=0.9E+13`

Either edit this value (under heading: # — WEATHERING —) or add a new line at the end of the user config file specifying the value you want. Units are mol of CaCO₃ weathered per year.

This parameter could also be adjusted to implicitly simulate the effect of a change in carbonate deposition in coral reefs and other shallow water carbonates, changes that GENIE cannot simulate explicitly. See Ridgwell et al. [2003]

(http://www.seao2.org/pubs/ridgwell_et_al_2003a.pdf) for references and discussion of the sort of change in carbonate deposition on the shelves that might have taken place. A decrease in CaCO₃ removal on the continental shelves can be simulated by increasing the weathering flux to the open ocean. In other words, you can look at the parameter `rg_par_weather_CaCO3` as representing the residual weathering flux to the open ocean, after some of the weathering flux has been removed in coastal areas. Even if global weathering of the continents did not change, any reduction in CaCO₃ precipitation and removal on the continental shelves would result in an increased solute flux to the open ocean.

[HINT: keeping track of how mean sediment wt% CaCO₃ changes will be helpful, as may 2D sediment distributions and ultimately, the sediment core records.]

13.2.2 Iron fertilization.

Read up on this first, e.g., see references in Kohfeld and Ridgwell [2009]. The glacial was dustier than present, hence there can only have been increased aeolian iron supply to the ocean surface. However, what is not so clear is how important (relative to Fe being upwelled) aeolian Fe is today, let alone during the last glacial ...

Anyway: one way to increase the aeolian Fe supply to the ocean surface is simply to increase the solubility of the Fe in dust. This is controlled by the parameter:

`bg_par_det_Fe_sol=0.0015`

with the default being a global average dust Fe solubility of 0.15. Increasing will increase the Fe input to the ocean surface everywhere (in direct proportion to the modern spatial pattern). The pattern of total aeolian Fe supply is recorded in the (2DBIOGEM) variable:

`misc_sur_fFe_tot_mol`, with the dissolved component under:

`misc_sur_fFe_mol` (`misc_sur_Fe_sol` is the map of solubility, which in GENIE is not uniform in space – any idea what the reason for this assumption might be?). (A glacially-explicit map of dust deposition could also be applied in place of the modern deposition map – if you would like to test this, I can create one, but note that there are very significant ‘errors’ in re-gridding dust maps to this highly simplified continental topography.)

[HINT: viewing maps of particulate organic carbon (POC) export may be particularly helpful.]

13.2.3 Remineralization depth.

There is no temperature control on the rate of bacterial degradation of sinking organic matter (see: book chapter + references therein) but the effect of lower ocean temperatures and a slower rate of bacterial degradation of organic matter can be simulated by specifying that particulate organic

matter reaches greater depth before being remineralized (and CO₂ and PO₄ released back to the seawater). The namelist parameter that controls the e-folding depth reached by particulate organic matter before remineralization is:

`bg_par_bio_remin_POC_eL1=589.9451`

Either edit this value (under heading: # — REMINERALIZATION —) or add a new line at the end of the user config file specifying the value you want. Units are m. Read Ridgwell et al. [2007] for additional discussion of this parameter. See Figure 2-4 in Ridgwell [2001] (http://www.seao2.org/pubs/ridgwell_thesis.pdf) for an illustration of how the flux of particulate organic matter decreases with depth in the ocean, plus references therein.

There is also an associated parameter: `bg_par_bio_remin_POC_frac2`, which sets a fraction of organic matter that is assumed to settling through the water column completely un-altered (currently assigned a value of 0.025 == 2.5%), but this is arguably less appropriate to change than the remineralization length-scale of the more labile fraction (97.5% of exported particulate organic carbon).

[HINT: viewing distributions of PO₄ and/or O₂ in the ocean may be helpful. Perhaps also d13C.]

13.2.4 Macro nutrient inventory and uptake.

Suggestions have been made that nutrients were used more efficiently during the LGM, meaning that for the same nutrient uptake at the surface more carbon was exported to depth in the ocean. See: Omata et al. [2006]. There are also a bunch of (relatively old) hypotheses concerning differences between glacial and modern ocean in how much nitrate (NO₃⁻) there was. There is no NO₃⁻ in this version of GENIE (just PO₄ 3- and Fe), but an analogous change can be made to the phosphorous cycle.

For the nutrient-to-carbon ratio in organic matter, the relevant parameter is:

`bg_par_bio_red_POP_POC=106.0`

To change the default value (106.0), add a new line at the end of the user-config file specifying the value you want. A larger number means that PO₄ is being utilized more efficiently and more organic matter is being produced for the same nutrient consumption.

If you would like to test the effect of adding more PO₄ to the (glacial) ocean – a forcing is provided, called:

`p0000b_FeMahowald2006_ADJUST_phosphate`

Note that adjusting the ocean PO₄ inventory should only be done one (and not accidentally in each successive experiment!).

[HINT: viewing distributions of PO₄ and/or O₂ in the ocean may be helpful. Also ocean sediment CaCO₃ distributions.]

13.2.5 CaCO₃:POC rain ratio.

Kicked off by a classic 1994 Nature paper by Archer and Maier-Reimer (see: Kohfeld and Ridgwell [2009]), one powerful means of changing atmospheric CO₂ that has been proposed involves changes in the export ratio between CaCO₃ (shells) and POC (particulate organic matter). Such a change in ratio could come about through a variety of ways (e.g., via the 'silica leakage hypothesis' (see: Kohfeld and Ridgwell [2009]) and also through the direct effect of Fe on diatom physiology (see Watson et al. [2000] in Nature and also Supplemental Information). There are also ideas about an opposite ocean acidification effect, whereby the less acidic glacial (compared to modern) ocean

led to increased calcification and CaCO₃ export. Note that this response (higher saturation == great calcification) is encoded into your model configuration – see Ridgwell et al. [2007b].

In GENIE, the CaCO₃:POC rain ratio is controlled (technically: scaled) by the parameter:
`bg_par_bio_red_POC_CaCO3=0.03`

The pattern of CaCO₃:POC rain ratio is not uniform across the ocean (why? (see: Ridgwell et al. [2007, 2009]), and its pattern can be viewed in the (2D BIOGEM) netCDF variable:
`misc_sur_rCaCO3toPOC`.

[HINT: viewing sediment CaCO₃ distribution may be helpful.]

13.2.6 Sea-ice extent.

Changes to sea-ice extent have already taken place due to changes in radiative forcing and planetary albedo (made previously). There is no much you can do to further adjust sea-ice extent, other than via further changes to climate (via radiative forcing and/or albedo).

13.2.7 Atlantic circulation.

There are a variety of ideas and hypotheses about glacial ocean circulation and what influence it had on atmospheric CO₂. At least with respect to making tests and experiments in models, a common ploy has been to produce a collapsed AMOC (e.g., see Chikamoto et al. [2008] (JGR 113)). Rather than apply a continuous freshwater forcing to the ocean throughout an extended (sediment interaction) time-scale (why would this not be a good idea?), there is a parameter in the model which creates an adjustment of the salt balance between the different ocean basins (to make the Atlantic more salty compared to the Pacific). (In other words: salt/freshwater is re-partitioned between the ocean basins rather than 'new' freshwater or salt externally added.) This parameter is:

`ea_28=0.726862013339996340`

Setting it to e.g., 0.0, will result in a collapsed AMOC. But maybe that is too extreme? (You might read up a little on the glacial ocean circulation literature and chose a value that gives as an appropriate change to Atlantic circulation as you can judge from the data and literature.)

[HINT: viewing distributions of PO₄ and/or O₂ and/or d¹³C in the ocean may be helpful. Also ocean sediment CaCO₃ distributions.]

13.2.8 Global ocean circulation / 'brine rejection'.

Some recent research has focussed on the possible role of 'brine rejection' in creating a saltier Antarctic bottom waters (e.g. see Adkins et al. 2002 Science paper) and hence a denser and more stratified deep ocean,, with the idea being this will trap carbon more efficiently. For a very recent study (and references therein), see:

<http://www.clim-past.net/6/575/2010/cp-6-575-2010.html>

GENIE has the capability to include this effect (at least crudely) and similarly to Bouttes et al. [2010]. For this, three namelist parameter values need to be set:

```
bg_ctrl_force_GOLDSTEInTS=.TRUE.  
bg_par_misc_brinerejection_frac=0.1  
bg_par_misc_brinerejection_jmax=9
```

The first, simply allows the BIOGEM biogeochem module to directly influence ocean circulation. The second is the fraction of salt, rejected during sea-ice formation (e.g., see Bouttes et al. [2010]) that is transferred directly to the bottom-most (underlying) ocean cell in the model. The

first sets a latitude limit (counted in cells) to the effect – a value of 9 will restrict brine rejection to the Southern Ocean; a value of 18 will allow it to take place in the North Atlantic as well. (Note that in e.g., Bouttes et al. [2010], the effect is considered only in the Southern Ocean.)

[HINT: viewing distributions of PO₄ and/or O₂ and/or d₁₃C in the ocean may be helpful. Also ocean sediment CaCO₃ distributions.]

MISC.

There are of course other possibilities for adjusting the model, although you need an a priori reason for doing so and what about the possible glacial state of global carbon cycling and climate you are trying to encapsulate. Examples might include wind speed (or air-sea gas exchange).

Even if you achieve atmospheric CO₂ of ca. 190 ppm (and actually, with some mechanisms on their own and also in combination, it is quite easy to achieve this), how do you know if you are ‘right’? Many of the important constraints are summarized in Kohfeld and Ridgwell [2009] and Archer et al. [2000]. In particular:

- The distribution of the CaCO₃ content of deep-sea sediments. e.g., see Figure 6 in Archer et al. [2000]. You are not ‘allowed’ to blanket the entire ocean floor with CaCO₃ if you want to be consistent with the paleoceanographic record of the LGM ;)
The predicted distribution of the CaCO₃ can be used to assess your circulation change – note that there is much less CaCO₃ in sediments in the North Atlantic at the glacial [Archer et al., 2000]. See: Chikamoto et al. [2008] for a model assessment of the impact of AMOC changes on deep-sea sediment composition.
- The ocean should not go ‘anoxic’ (i.e., little to no dissolved oxygen left) over large expanses. (But you might consider this relative to the modern configuration – i.e., should the modern simulation under-estimate oxygen concentrations in the deep ocean, so will the glacial simulation, even if you get the mechanisms exactly ‘right’.)
- There is a map of estimated changes in the biological flux to the ocean floor in Kohfeld and Ridgwell [2009] (also read the original reference). In the 2D netCDF file, the variable focnsed_POC gives you the flux of particulate organic matter (actually, carbon) to the ocean floor. By constructing a difference map of your glacial-interglacial predicted changes, you could contrast directly to the Kohfeld et al. [2005] reconstruction.
- The GENIE model is set up to predict d₁₃C distributions. See: Curry and Oppo [2005]. There is also an atmospheric record of d₁₃C (also predicted by GENIE) – see: Smith et al. [1999] and a more recent paper in GBC: Lourantou et al. [2010] (‘Constraint of the CO₂ rise by new atmospheric carbon isotopic measurements during the last deglaciation ’).
- Other proxies offer varying constraints at the global or regional scales. e.g., see: Elderfield and Rickaby [2000] (Cd/Ca ratios).

Note that commonly in (glacial CO₂) modeling studies, a steady state (or quasi steady state) simulation is run for the glacial (and compared to pre-industrial). The version of cGENIE you have is sufficiently fast to do this quite effectively. It is possible to do non-state (glacial-interglacial) simulations, e.g. Ridgwell [2001], but this is rather more involved.

Also note that in all of the above possible adjustments to the global carbon cycle, the mechanism of carbonate compensation is operating. Hence there will be direct (changes in carbon cycling within the water column) and indirect (interaction between ocean and deep-sea sediments) processes operating that will affect CO₂. Carbonate compensation will typically take a few 10s of

thousands of years to fully adjust atmospheric CO₂. Not all previous modeling studies include this effect and in some cases it can drastically influence the predicted change in atmospheric CO₂.

Finally ... maybe you have achieved close to 190 ppm and are not unreasonably consistent with various paleoceanographic proxies and are hence feeling rather pleased with yourself ... O – sorry – I forgot to mention a little something:

- There is an approximately 3% (1 PSU) increase in salinity (and other dissolved tracers) due to the presence of large (Northern Hemisphere) ice sheets, and hence loss of freshwater from the ocean and lower sea-level associated with the last glacial.

Ahhh ... and I also forgot:

- It is thought that there was a release of carbon stored on land in vegetation and soils during glacial climates. (cGENIE has some capabilities to model changes in terrestrial carbon storage and hence predict this, but you are not using a version with this science module enabled.)

Unfortunately – both these effects act to increase atmospheric pCO₂ at the last glacial (or decrease it across the deglacial transition, which ever way around you like to think of it). So you are actually rather further off achieving a net 90 ppm difference than you thought.

13.2.9 Salinity

To test the effect of a 3% increase in salinity at the last glacial, you need to create a new userconfig file (copied from your last glacial ‘best guess’ configuration), add the lines:

```
bg_ctrl_force_GOLDSTEInTS=.true.  
bg_par_forcing_name='p0000b_FeMahowald2006_ADJUST_salinity'
```

and run using your last glacial best guess as the restart.

If you want to check that you have applied this forcing correctly: Mean ocean salinity should end higher than the preindustrial restart that was originally provided (or compared to your best guess glacial run). The file biogem_series_ocn_sal.res is the time-series results file for ocean salinity – the 2nd column is the mean ocean salinity. Originally it was 34.904 PSU (or o/oo), now it should be about 35.9.

Note how atmospheric pCO₂ has responded to the change in ocean volume and sea-level (and tracer concentrations) alone. How does this reduced resolution version of cGENIE compare to published estimates (too much; too little; why? or if

13.2.10 Terrestrial carbon storage

For the reduction in terrestrial carbon storage, you need to try a further experiment (on top of and/or following on from the salinity change experiment), with the user-config line:

```
bg_par_forcing_name='p0000b_FeMahowald2006_ADJUST_terrestrialC'
```

These two forcings are effectively one-off changes imposed on the global climate and carbon cycle. By selecting the salinity forcing, you add 1 PSU of salinity to the entire ocean (and concentrate proportionally all dissolved tracers in the ocean) in a single year. Obviously you only want to do this once, not multiple times (otherwise you will get an increasingly salty ocean ...). Similarly with the terrestrial carbon change – as specified, this forcing results in 500 PgC of carbon being added to the atmosphere over a period of 500 years (i.e., at a rate of 1 PgC yr⁻¹) as if to simulate a commensurate reduction in carbon stored on land. One strategy might be to implement this as a

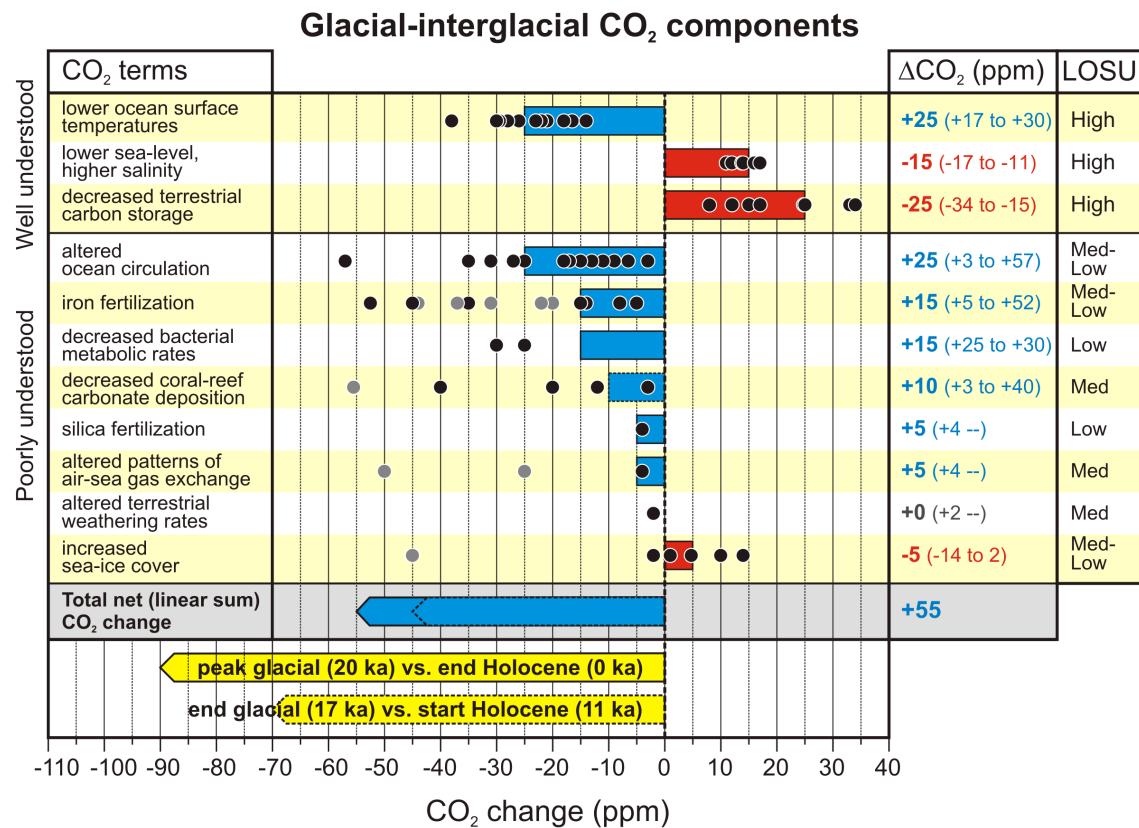


Figure 13.1: Baaa.

second phase of (additional) spin-up (after the salinity modification). Note that while the magnitude of the glacial-interglacial salinity and sea-level change is well constrained, that of the terrestrial biosphere is not (e.g., see: Kohfeld and Ridgwell [2009]). In investigating the potential causes of low glacial CO₂, do not feel constrained to necessarily run with the default (500 PgC) forcing provided ... (Think for yourselves!)

To confirm that you have correctly added (rather than subtracted!) carbon to the ocean+ atmosphere – the ocean + atmosphere carbon inventories should start changing from the start of the experiment incorporating the carbon change forcing

(p0000b_FeMahowald2006_ADJUST_terrestrialC)

and the change should be approximately uniform. You can calculate the change in ocean + atmosphere carbon inventory from the atmospheric CO₂ time-series file (biogem_series_atm_pCO2.res – column #2 is the global CO₂ inventory in mol) and the ocean total dissolved carbon timeseries file (biogem_series_ocn_DIC.res – column #2 is the global DIC (total dissolved inorganic carbon) inventory in mol). Note you will have to convert from mol to gC (or PgC) in order to compare to the amount you requested. If the rate of inventory change turns out to be not quite linear, and particularly if the inventory change should turn to be not quite what you were expecting ... why? (Hint: refer to the mechanisms discussed in the lecture (and papers) relating deep-sea sediments and weathering to changes in total carbon (e.g., fossil fuel CO₂ release.)

Overview (and types of model output)

Time-slice output

Frequency and timing of *time-slice* data saving

Experiment end saving

Seasonal/monthly data saving

More frequent data saving

Time-series output

Frequency and timing of *time-series* data saving

Saving orbital insolation

Data field selection

Re-start files

Useful output

Physics

(Bio)Geochemistry

Biology/Ecology

14. muffin model output

This section covers **muffin** saves data and how to ensure that the variables you want are saved and when you want.

- Overview.
- *Time-series* output.
 - *Time-series* file naming conventions.
 - Specifying frequency and timing of *time-series* data saving.
 - Seasonal/monthly data saving.
- *Time-slice* output.
 - *Time-slice* file naming conventions.
 - Specifying frequency and timing of *time-slice* data saving.
- Specifying which data fields to be saved in the *time-series* and *time-slice* format.
- *Re-start* files.

14.1 Overview (and types of model output)

The results of experiments are written to the directory:

```
~/cgenie_output
```

For any particular experiment, all saved model results, plus copies of input parameters and the model executable, are gathered together in a directory that is assigned the same name as the experiment (== *user-config* file name), e.g.:

```
EXAMPLE.worbe2.Ridgwelletal2007.SPIN
```

Every science module saves its results in its own individual sub-directory within the experiment directory. So for the module that calculates ocean biogeochemical cycles – **BIOGEM**, the results files will thus be found in:

```
~/cgenie_output/EXAMPLE.worbe2.Ridgwelletal2007.SPIN/biogem
```

Note that **ATCHEM** does not save its own results (**BIOGEM** can save information about atmospheric composition and air-sea gas exchange) while **SEDGEM** essentially saves results only at the very end of a model experiment (**BIOGEM** can also save the spatial distribution of sediment composition as time-slices as well as mean composition as a time-series). Furthermore, in order to attain a common format for both ocean physical properties and biogeochemistry, **BIOGEM** can save a range of ocean results in addition to temperature and salinity, such as: velocities, sea-ice extent, mixed layer depth, convective frequency, etc. Also note that **SEDGEM** saves data only at the end of an experiment¹.

Saving full spatial distributions for any or all of the tracers at each and every time-step is not practical, not only in terms of data storage but also because of the detrimental effect that repeated disk access has on model performance. Instead, **BIOGEM** saves the full spatial distribution of whatever tracer, flux, and/or physical properties of the system are required (how what fields are required is specified is discussed later), only at one or more predefined time points (in years). These are called '*time-slices*'. However, rather than taking an instantaneous snapshot, the time-slice is constructed as an average over a specified integration interval. The second main data format for model output is that of a '*time-series*' of change in a single (integrated) property of the Earth system. Model characteristics must be reducible to a single meaningful variable for this to be practical (i.e., saving the time-varying nature of 3-D ocean tracer distributions is not). Suitable reduced indicators include: the total inventories in the ocean and/or atmosphere of various tracers (or equivalently, the mean global concentrations / partial pressures, respectively), global sea-ice coverage. Like *time-slices*, the data values saved in the *time-series* files represent averages over a specified integration interval (one year by default). For both *time-slices* and *time-series* output, the files themselves are created during model initialization and are periodically updated (appended to) during the experiment. Hence, even before the experiment has finished they may contain data that is useful to view and can be used to check on the progress of an experiment.

ATCHEM

In the **ATCHEM** results directory, only the following file will be present:

1. `_restart.nc` – *Re-start* file – a snap-shot of the 2D distribution of atmospheric composition at the very end of the experiment. Not intended for user-access, although it can be plotted just like any normal *netCDF* format file.

¹With the exception of sediment core location environmental properties, which are saved more frequently.

BIOGEM

For **BIOGEM**, some or all of the following files will be present:

1. `_restart.nc` – *Re-start* file – a snap-shot of the 3D distribution of biogeochemical properties of the ocean at the very end of the experiment. Not intended for user-access, although it can be plotted just like any normal *netCDF* format file.
2. `fields_biogem_2d.nc` – 2-D fields of (mostly) ocean bottom, ocean surface, and sediment surface properties.² Also: water-column integrals of certain geochemistry diagnostics, air-sea gas exchange fluxes, atmospheric composition.
3. `fields_biogem_3d.nc` – 3-D fields ocean dissolved and particulate tracer properties.³
4. `biogem_series_* .res`⁴ – *Time-series* results files – globally and surface-averaged (and sometimes also benthic (bottom) surface averaged) property values as a function of time in plain text (ASCII) format.
5. `biogem_year_*_diag_GLOBAL.res` – Miscellaneous global diagnostic information. It is saved at each requested time-slice with the file-name string containing the mid-point of the time-slice (as years). The diagnostics include:
 - time mid-point and integration interval
 - global ocean surface area and volume
 - mean global air-sea gas exchange coefficient (for CO₂)
 - mean atmospheric tracer concentrations plus total inventory
 - mean ocean tracer concentrations plus total inventory
 - mean plus total global productivity
 - mean plus total global sedimentation

SEDGEM

In the **SEDGEM** results directory, some or all of the following files will be written:

1. `_restart.nc` – *Re-start* file – a snap-shot of the 2D distribution of sedimentary properties at the very end of the experiment. Not intended for user-access, although it can be plotted just like any normal *netCDF* format file.
2. `fields_sedgem_2d.nc` – Contains 2-D fields of sediment surface and ocean bottom properties.⁵
3. `sedcore.nc` – *netCDF* format file containing the stacked records of accumulated deep-sea sediment composition.

The locations (if any) of sediment cores to be saved is specified in a plain text (ASCII) file pointed to by the string value of the *namelist* parameter `sg_par_sedcore_save_mask_name`⁶. In the mask file, a '1' indicates a location to save a sediment core at, and a '0' indicates that no sediment core should be saved at this location. This file must be present, so to save no sediment cores, simply populate the file with all zeros in an xx by yy grid.

²The mid-points at which time-slices are saved are specified as described above.

³The mid-points at which time-slices are saved are specified as described above.

⁴.res is a useful format for processing in **MATLAB**; for other programs, other extensions are needed. If using the Mathematica data processing scripts - see `genie-docs/cGENIE.AutomationScripts` - .dat is needed; this can be set with `gm_string_results_ext=".dat"`

⁵This data is saved only at the termination of an experiment (i.e., the *netCDF* file contains only a single time-slice).

⁶The location of this file is specified by the **SEDGEM** data input directory namelist parameter: `sg_par_indir_name` which by default is `~/genie-sedgem/data/input`.

4. `sedcoreenv_*` – These files contain pseudo time-series of surface sediment environmental properties at each of the requested sediment core locations (if any are chosen).
5. `seddiag_misc_DATA_GLOBAL.res` – A summary of mean global sedimentation, dissolution, and preservation fluxes, and surface sediment composition.
6. `seddiag_misc_DATA_FULL.res` – Surface sediment and bottom water properties at each and every sediment grid point.

ROKGEM

In the **ROKGEM** results directory, some or all of the following files will be written:

1. `fields_rokgem_2d.nc` – 2-D fields of (mostly) land surface, ocean surface, and atmospheric properties related to weathering.
2. `biogem_series_*` – Time-series results files.

14.2 Time-slice output

14.2.1 Frequency and timing of time-slice data saving

Rather than taking an instantaneous snapshot, the time-slice is averaged over a specified integration interval Δt (in years), defined by the parameter `bg_par_data_save_slice_dt`⁷. The model state is thus integrated from time $t_n - \Delta t/2$ to $t_n + \Delta t/2$. For instance, setting a value of $\Delta t = 1.0$ year results in all seasonal variability being removed from the saved time-slices, and successive time-slices then only reflect long-term (>1 year) trends in system state.

The mid-point years (t_n) for which time-slices should be saved are specified in a single column plain text (ASCII) file in the `cgenie.muffin/genie-biogem/data/input` directory, whose name is specified by the parameter `bg_par_infile_slice_name`⁸. For example, the default *time-slice* specification file `save_timeslice.dat` contains the specification⁹:

```
-START-OF-DATA-
0.5
1.5
4.5
9.5
19.5
49.5
99.5
199.5
499.5
999.5
1999.5
4999.5
9999.5
19999.5
49999.5
99999.5
199999.5
499999.5
999999.5
-END-OF-DATA-
```

where `-START-OF-DATA-` and `-END-OF-DATA-` are simply tags delineating the start and end of the time point data. Use of this particular specification lends itself to simple experiment run durations to be adopted (e.g., 10, 100, 10000 years). It provides a good generic starting point in that save frequency is faster to begin with (when environmental variables are more likely to be rapidly changing) and less frequently later (when environmental variables are unlikely to be changing rapidly and maybe converging to steady-state).

To change the time points used for *time-slice* data saving, either direct edit this file (less good), or create a new file (e.g. simply copy and rename `save_timeslice.dat`) with the required save

⁷An empty list is valid - time-slices will then be populated for you at an interval set by the time-slice integration interval. But if you really don't want any time-slices, just set the first (or only) time point to occur beyond the end year of the run.

⁸The location of this file is specified by the **BIOGEM** data input directory parameter: `bg_par_indir_name` which by default is `~/genie-biogem/data/input`.

⁹The order in which the time sequence is ordered (i.e., ascending or descending time values) does not actually matter in practice as long as the list of times is ordered sequentially. The list will be internally re-ordered if necessary according to the selection of 'BP' (the model running backwards-in-time) or not according to the logical value of the parameter `bg_ctrl_misc_t_BP`, which is `.false.` by default.

frequency and timing and saved to the `cgenie.muffin/genie-biogem/data/input` directory, with the parameter `bg_par_infile_slice_name` pointing to the new filename).

14.2.2 Experiment end saving

Just in case an experiment run duration is chosen such that there is no corresponding save point anywhere near the end of the run, a *time-slice* is automatically saved at the very end of an experiment regardless of whether one has been specified or not and with the same averaging as used for the specified *time-slices*.

14.2.3 Seasonal/monthly data saving

Time-slice (but not currently *time-series*) data can be saved seasonal or even monthly by selected by setting a single parameter rather than e.g. specifying a monthly or seasonal data save interval and editing the time-slice definition file with a series of min-points for months (or seasons). The way it works is that the overall averaging interval (parameter: `bg_par_data_save_slice_dt`)¹⁰ is broken down into sub-intervals of averaging. i.e., breaking down a year interval (the default) into 4 will give seasonal averaging. The parameter: `bg_par_data_save_slice_n` where n sets the number of time steps in each sub-interval of data saving and hence determines whether the averaging is e.g. seasonal or monthly. The slightly tricky part is to be sure of how many time steps in each year ;)

By default, **cGENIE.muffin** employs 96 time-steps per year for a 16-level ocean circulation model (**GOLDSTEIN**) and 48 for **BIOGEM**¹¹. Hence for a 16-level ocean configuration, seasonal data saving would be obtained with:

```
bg_par_data_save_slice_n=12
```

(12 **BIOGEM** steps per averaging interval out of a total of 48), and monthly averages with:

```
bg_par_data_save_slice_n=4
```

(i.e. 4 **BIOGEM** steps for each of the 12 monthly averaging intervals, giving of a total of 48).

For lower resolution configurations of **cGENIE.muffin**, **GOLDSTEIN** may be operating on 48 time-steps per year, and **BIOGEM** on 24 or even 12. As **cGENIE.muffin** starts up it will report the ocean and biogeochemical time-stepping, such as:

```
>> Configuring ...
Setting time-stepping [GOLDSTEIN, BIOGEM:GOLDSTEIN]: 100 2
```

which specifies 100 time-steps per year for **GOLDSTEIN**, and 50 per year (100/2) for **BIOGEM** for the case of a 16 level ocean using the `runmuffin.t100.sh` run script. Note that for every year mid-point specified in the *time-slice* specification file, 4 or 12 (for seasonal and monthly, respectively) times as many time-slices will actually be saved.

¹⁰Default value = 999

¹¹Note that when running using `runmuffin.t100.sh`, 100 time-steps are taken in the ocean and 50 in **BIOGEM** for a 16 level ocean model.

14.2.4**More frequent data saving**

Explicit frequent saving of fields or properties at specific locations can be done by setting a more higher save frequency of the time-slice data. However, because the 2D and 3D fields may contain a variety of unwanted variables in addition to the target one, save frequency is likely to be limited by the maximum netCDF file size that can sanely be manipulated. The practical maximum is around 100, depending on the number of types of data field selected to be saved. Several alternative options are available:

- (trivial) Make do with global or surface (or benthic) means in the time-series output.
- Cut down the types of data saved to the absolute minimum (see 'Data field selection' below).
- Save only 2D data. This can be accomplished by setting the parameter:

```
bg_ctrl_data_save_2d=.true.  
bg_ctrl_data_save_3d=.false.
```

(both are `.true.` by default). This disables the 3D data saving, although an empty netCDF file will still be created.

- Save the tracer fields in 3D, but at the save frequency of time-series data saving¹². This can be done by setting:

```
bg_ctrl_data_save_3d_sig=.true.  
(by default, .false.).
```

¹²This is in addition to normal 3D saving at the time-slice data saving frequency

14.3 Time-series output

14.3.1 Frequency and timing of time-series data saving

For results *time-series*, a file containing a series of model times (t_n) at which data points need be saved is defined in the same way as for *time-slices*, with the filename specified by the parameter `bg_par_infile_sig_name`. Again, the data values saved in the time-series file do not represent discrete values in time but an average, calculated from time $t_n - \Delta t/2$ to $t_n + \Delta t/2$ as per the construction of time-slices. The averaging interval, Δt , is set by the value of the parameter `bg_par_data_save_sig_dt`. The format is also identical to before (with tags delineating the start and end of the list of mid-points). If there are less than two elements present in the list, a default frequency of data saving will be invoked, set equal the averaging interval, except in the situation that this results in an unreasonably large amount of data, when an order of magnitude (or more than one order of magnitude where necessary) fewer save points are assumed.¹³ The default setting:

```
bg_par_infile_sig_name='save_timeseries.dat'
```

provides for reasonably generic data saving, with the save frequency faster to begin with and becoming progressively less frequently later.

There is a related facility to `bg_par_infile_sig_name` for **SEDGEM** and **ROKGEM** in the parameter: `xx_par_output_years_file_0d`, where `xx` is `sg` for **SEDGEM** and `rg` for **ROKGEM**. These specify files in the `genie-*gem/data/input` directory and again contain a list of years for 0D (time-series) output to be generated at. However, unlike **BIOGEM**, the data saved *do* represent discrete values in time and *not* e.g. annual averages.

14.3.2 Saving orbital insolation

[see orbits HOW-TO]

¹³For historical reasons ... the maximum number of time-series (and time-slice) data points was set to 4096. This is set by the parameter `n_data_max` in `biogem.lib.f90` and can be altered if required.

14.4 Data field selection

Model output – both *time-slice* and *time-series* data are saved in blocks or by categories of model variables. For instance, all dissolved tracers in the ocean (3D netCDF *time-slice* and/or *time-series*), or all particle flux fields, all carbonate chemistry and associated variables, all surface sediment composition, etc etc. This still requires a multitude of parameters, one for each category and generally also one for each of *time-slice* and *time-series* data. In an attempt to simplify this, a single parameter, `bg_par_data_save_level`, specifying the sort and amount of data to save can be set instead.

The value of the `bg_par_data_save_level` save level parameter is given as an integer between 0 and 99, and has the following effect:

- 0 – Save nothing.
- 1 – Minimum – basic geochemistry only, i.e. ocean and atmosphere tracer fields (omitting e.g. the miscellaneous fields and time-series – see below).
- 2 – Basic output == basic geochemistry and physics.
- 3 – Basic + biology diagnostics, i.e. ocean and atmosphere tracer fields plus particulate flux fields and biological diagnostics such as limitations on export.
- 4 – Basic + geochemistry diagnostics, including output on air-sea gas exchange, ocean carbonate chemistry, and geochemical diagnostics such as remineralization rates and transformation, ocean pH field (3D), Fe cycle and speciation diagnostics (2D). In conjunction with the **ROKGEM** module, also: weathering fluxes.¹⁴
- 5 – Basic + biology + geochemistry diagnostics. A combination of 3 and 4.
- 6 – Basic + tracer diagnostics. Tracer diagnostics includes: N*, P* etc., water column inventories (2D).
- 7 – Basic + tracer + proxy diagnostics. Proxy diagnostics includes: ocean surface and benthic (and surface-benthic) tracers (2D). Also trace metals (e.g. Cd).
- 8 – Basic output + biology + geochemistry + tracer + proxy diagnostics.
- 9 – Basic output + full physics (e.g. all grid specifications and properties).
- 10 – Ocean acidification option == basic geochemical output fields plus all carbonate chemistry.
- 99 – Save everything.
- >99 – Use explicit user-specified settings for individual save categories. This is the default and is broadly consistent with previous version of the model.

All of options 2-11 save as *time-slices*:

- atmosphere tracer fields (2D)
- ocean tracer fields (3D)
- various miscellaneous diagnostics, including: ocean velocity (3D), overturning streamfunction (2D), sea-ice extent and thickness (2D), incident radiation (2D), convection diagnostics (2D), air-sea gas exchange diagnostics (2D).
- core-top sediment composition fields (2D) (if **SEDGEM** is selected)

and as *time-series*:

- atmosphere tracer properties (as: atmospheric inventory (in mol) and concentration ($mol kg^{-1}$) or isotopic composition)
- ocean tracer properties (as: ocean inventory, plus mean (whole) ocean, or isotopic composition)

¹⁴The most common option.

- mean surface and benthic tracer properties
- various miscellaneous diagnostics, including: insolation, sea-ice extent, volume, and thickness; global overturning stream-function, ocean surface pH, land surface temperature, and Fe parameters.
- sediment (core-top) composition data (if **SEDGEM** is selected)

In addition, further output will be automatically added to the suite of saved data depending on the module selected and also for certain sorts of *forcing*.

14.5 Re-start files

Re-start files are saved in the results directories of each module. For **ATCHEM**, **BIOGEM**, and **SEDGEM**, these are in *netCDF* format.

For the climate modules of **cGENIE.muffin** (**GOLDSTEIN**, **GOLDSTEIN-SEAICE**, **EMBM**), *re-start* files can be selected to be saved in either plain text (ASCII) or *netCDF* format. ASCII format is the current default.

14.6**Useful output**

What follows is a short summary of some of the output and how it can be used.

Note – depending on the specific model configuration (which model modules are selected) and selected tracers, as well as specific output choice, not all these variables and files will be present in the model output.

14.6.1 Physics

Filename	Data	Application
biogem_series_*.res		
atm_humidity	Mean surface humidity. (?)	(rarely used)
atm_temp	Mean surface temperature. (degrees C)	Climate and climate change, when a simple global diagnostic is needed.
misc_opsi	Global minimum (-ve) and maximum (+ve) overturning streamfunction, also reported for Atlantic and Pacific basins. Units of Sv. (For certain modern configurations.)	Simple diagnostic of large-scale ocean circulation. There is some relationship of the maximum (negative and positive) overturning to ocean ventilation.
misc_seaice	Sea-ice fractional cover (%), thickness (m), and volume (m3).	As a simple climate (change) diagnostic.
misc_SLT	Mean global surface land temperature. (C)	For calibrating and analysing global weathering rates.
ocn_sal	Mean global, and typically also surface and benthic, ocean salinity. (PSU)	For characterizing freshwater changes and salinity <i>forcing</i> impacts.
ocn_temp	Mean global, and typically also surface and benthic, ocean temperature. (degrees C)	Climate and climate change.

Table 14.1: Summary of the main (useful) *time-series* output for climate-only ('physics') investigations.

variable	variable (long name)	Description	Application
atm_humidity	specific humidity	surface humidity (?)	(rarely used)
atm_temp	surface air temperature"	surface air temperature (degrees C)	climate patterns and anomalies, comparison with terrestrial temperature proxies
grid_mask	land-sea mask	land-sea mask (n/a)	copy-paste-edit to create masks for data analysis
grid_topo	ocean depth	ocean depth (m)	
ocn_sur_sal	surface-water sal	surface ocean salinity (PSU)	diagnosing freshwater forcing impacts, regions of likely deep-water formation
ocn_sur_temp	surface-water temp	surface ocean temperature (degrees C)	diagnosing ocean circulation patterns, pole-to-equator temperature gradients, surface ocean temperature proxy comparisons
ocn_ben_temp	bottom-water temp	benthic temperature (C)	diagnosing ocean circulation patterns, benthic temperature proxy comparisons
phys_cost	convective cost	rate of convective adjustments anywhere in the water column (n/a)	diagnosing deep mixed layers (and light limitation of biology) and deep-water formation regions
phys_opsi	Global streamfunction	global overturning stream-function (Sv)	diagnosing large-scale ocean circulation patterns, sources of deep-water formation, deep ocean ventilation
phys_psi	Barotropic streamfunction	barotropic stream-function (Sv)	diagnosing wind-driven ocean circulation patterns, effect of gateways
phys_seaice	sea-ice cover (%)	sea-ice cover (%)	(climate / sea-ice)
phys_seaice_th	sea-ice thickness	sea-ice thickness (m)	(climate / sea-ice)

Table 14.2: Summary of the main (useful, physics-focussed) 2D time-slice output.

variable	variable (long name)	Description	Application
ocn_sal	salinity	ocean salinity (PSU)	diagnosing ocean circulation patterns
ocn_temp	temperature	ocean temperature (C)	diagnosing ocean circulation patterns
phys_u	ocean velocity - u	Eastwards component of ocean velocity (m/s)	diagnosing ocean circulation patterns and currents
phys_v	ocean velocity - v	Northwards component of ocean velocity (m/s)	diagnosing ocean circulation patterns and currents
phys_w	ocean velocity - w	upwards component of ocean velocity (m/s)	diagnosing ocean circulation patterns (note that velocity is measured at the top of an ocean depth layer, hence n/a for the surface layer)

Table 14.3: Summary of the main (useful) 3D time-slice output.

14.6.2 (Bio)Geochemistry

Filename	Data	Application
biogem_series_*.res		
atm_pcO ₂	Global inventory (<i>mol</i>), mean concentration (<i>atm</i>) of atmospheric <i>CO</i> ₂ .	Drivers of and feedbacks with climate. Diagnostic of response to carbon emissions (and removal).
atm_pcO ₂ _13C	¹³ C inventory (<i>mol</i>) and $\delta^{13}\text{C}$ of atmospheric <i>CO</i> ₂ .	Diagnostic of carbon emissions (and removal). Comparison with (terrestrial) proxy $\delta^{13}\text{C}$ data.
atm_pO ₂	Global inventory (<i>mol</i>), mean concentration (<i>atm</i>) of atmospheric <i>O</i> ₂ .	Limited use. Checking on <i>restoring forcing</i> of atmospheric <i>O</i> ₂ . Impacts of <i>C</i> _{org} burial if included in model.
carb_sur_conc_*	Carbonate chemistry components (mean surface) (mol kg^{-1}).	Not generally useful.
carb_sur_H	Surface ocean mean $[H^+]$ (mol kg^{-1}).	More useful is pH – reported under misc (see below).
carb_sur_ohm_arg	Mean surface aragonite saturation.	Ocean acidification impacts of <i>CO</i> ₂ release. Weathering impacts. Relates to carbonate production by (modern) corals, pteropods.
carb_sur_ohm_arg	Mean surface calcite saturation.	Ocean acidification impacts of <i>CO</i> ₂ release. Weathering impacts. Carbonate production by foraminifera and coccolithophorids.
diag_misc_specified_forcing_*	Applied flux <i>forcings</i> (mol yr^{-1}).	Whenever a <i>restoring</i> , or <i>flux forcing</i> is specified, the actual flux employed, is saved here. Useful for diagnosing the flux associated with a restoring forcing (e.g. allowing emissions flux associated with RCP (<i>restoring forcing</i>) scenario to be diagnosed.)
fexport_CaCO ₃	Total flux (mol yr^{-1}) and flux density ($\text{mol m}^{-2} \text{yr}^{-1}$), of <i>CaCO</i> ₃ export from the ocean surface.	Carbonate production. Impacts of ocean acidification.
fexport_POC	Total flux (mol yr^{-1}) and flux density ($\text{mol m}^{-2} \text{yr}^{-1}$), of <i>POC</i> export from the ocean surface.	Particulate organic matter export. Impacts of changes in nutrient supply and limitation.
misc_surrH	Mean surface <i>pH</i> .	Ocean acidification.
ocn_DIC_13C	Global inventory (<i>mol</i>), mean global, surface, and benthic $\delta^{13}\text{C}$.	Carbon release and removal. Surface-benthic – indicator or strength of carbon export and the biological pump, as well ocean ventilation.
ocn_O ₂	Global inventory (<i>mol</i>), mean global, surface, benthic concentrations (mol kg^{-1}) of <i>O</i> ₂ .	Indication of changes in ocean anoxia. (long-term) Imbalances between burial and weathering.
ocn_PO ₄	Global inventory (<i>mol</i>), mean global, surface, benthic concentrations (mol kg^{-1}) of <i>PO</i> ₄ .	Nutrient limitation. (long-term) Imbalances between burial and weathering.
ocn_TDFe	Global inventory (<i>mol</i>), mean global, surface, benthic concentrations (mol kg^{-1}) of dissolved <i>Fe</i> .	Nutrient limitation. (long-term) Imbalances between burial and weathering.

Table 14.4: Summary of the main (useful, plus notes on a few less used) *time-series* output for (bio)geochemistry (non ecological) investigations.

variable	variable (long name)	Description	Application
atm_*		Distributions of gases and isotopes.	Not useful as the atmosphere is well-mixed. The <i>time-series</i> outputs are simpler and more useful.
carb_ben_ohm_arg carb_sur_ohm_cal		Benthic aragonite and calcite saturation.	Impacts of ocean acidification of distribution of benthic organisms. Indicator of sediment preservation.
carb_sur_ohm_arg carb_sur_ohm_cal		Ocean surface aragonite and calcite saturation.	Impacts of ocean acidification of distribution of planktic organisms.
fseaair_pCO ₂	pCO ₂ : net sea->air gas exchange flux density	Air-sea CO ₂ gas exchange.	Indicator of air-sea gas disequilibrium, regions of out-gassing/in-gassing.
misc_pH	ocean pH	Ocean surface pH.	Ocean acidification.
misc_sur_rCaCO ₃ toPOC	CaCO ₃ to POC export rain ratio	Particulate CaCO ₃ : POC export ratio from ocean surface.	Ocean acidification impacts.
ocn_sur_TDFe	surface-water TDFe	Ocean surface total dissolved Fe (mol kg ⁻¹).	Patterns of nutrient uptake and limitation.
ocn_sur_TDL	surface-water TDL	Surface ligand concentrations (mol kg ⁻¹).	(Stabilizes dissolved Fe, but so not useful itself.)
ocn_sur_PO4	surface-water PO4	Ocean surface [PO ₄] (mol kg ⁻¹).	Patterns of nutrient uptake and limitation.
ocn_ben_PO4	bottom-water PO4	Benthic [PO ₄] (mol kg ⁻¹).	Indicator of large-scale ocean circulation and ventilation.
ocn_ben_DIC_13C		Benthic δ ¹³ C.	Indicator of large-scale ocean circulation and ventilation. Model-data δ ¹³ C proxy comparison.
ocn_int_DIC	DIC water-column integrated tracer inventory	Pattern of water column integrated ocean DIC (i.e. dissolved carbon storage) (mol m ⁻²).	Indicator of CO ₂ emissions storage and transport when used in difference/anomaly maps and calculations.

Table 14.5: Summary of the main (mostly useful) 2D time-slice output for (bio)geochemistry.

14.6.3 Biology/Ecology

[Plotting with Excel](#)
[Plotting with Panoply](#)
Issues with Panoply default settings
Basic plots – examples
Difference (anomaly) plots
Ocean velocity plots

MATLAB plotting

[MATLAB 101](#)
[MATLAB and ASCII \(time-series\)](#)
[MATLAB 'Import Data' ...](#)
[MATLAB and netCDF \(time-slice\)](#)

15. Introduction to model results analysis

The primary format for saving spatial (2- and 3-D) data is *netCDF* (network Common Data Form). More information on the netCDF format and the libraries necessary to compile the model can be found [here](#). The writing of netCDF follows roughly the [CF1.0 convention](#) (NetCDF Climate and Forecast (CF) Metadata Convention). The netCDF output is written for the **BIOGEM** and **SEDGEM** modules separately, and both modules have a flag that suppresses spatial data file saving in ASCII format, with netCDF format being the default.

Under unix/linux, netCDF files can be interrogated with: `$ ncdump -h filename.nc` which will give you the header information of the file. The command is included in the netCDF library which has to be present to run the model anyway. It's useful to get the NCO software package helping to concatenate files or extract variables as shell command. A full list of available software to manipulate or graphically illustrate netCDF files can be found [here](#).

This Chapter covers how to visualize **muffin** output, with a particular emphasis on (netCDF) spatial data, via:

- **Panoply**

If you really really must insist on using Windozzzz, the recommended viewer for netCDF is [Panoply](#) (see below). **Panoply** can be run under linux and on the Mac (OS X).

There is also [ncBrowse](#). Again, this will also run under LINUX and on the Mac (OS X).

- **MUTLAB**

You can also view netCDF files using **MUTLAB**, for which a number of plotting functions are provided. An advantage here is that the **MUTLAB** code can be hacked to produce much more powerful and bespoke analysis and plots.

15.1 Plotting with Excel

Just ... don't do it!¹

¹Obviously, there may be circumstances where it might be helpful to plot time-series output in **Excel**, although in practice, time-series output is much easier and faster to load in and plot with **MATLAB**.

15.2 Plotting with Panoply

In the following sub-sections are some pointers and examples of **Panoply** plotting.²

When you open a netCDF file, you will be presented with a Datasets window (on the left hand side of the application window). This contains a list of all the variables available that you can display. You will find that the 'Long Name' description of the variable will be the most helpful to identify the one you want. Clicking (once) and highlighting an entry will display further information about that variable in the Variable window on the left hand side of the application window.

In a drop-down box at the bottom of the application window is an option for shortening the list of displayed variables in the netCDF file:

- Georeferenced variables – All the spatial (2 or 3D) variables.
- Plottable variables – As above, but now including axis definitions (which can be plotted as 1D lines).
- All variables – all of the variables.

To create a plot of a variable – simply double-click anywhere on the line containing that variable. A dialogue box will open with various options (Figure 15.1). For 3-D model output, you have the option of whether you want a 'Lat-Vert', 'Lon-Lat', or 'Lon-Vert' plot (for the 2-D fields, the only choice is 'Lon-Lat').

- 'Lat-Vert' plots – a common way of visualizing vertical distribution in the ocean. The default is for **Panoply** to display an average of all longitudes in a zonal mean. Un-ticking the Avebox in the Arraytab will enable you to specify a specific longitude section. Be aware that Panoply likes to plot the depth inverted by default ...
- 'Lon-Lat' plots – the classic 2D, top-down view of the ocean. There are multiple levels (depth layers) in the ocean of data that can be plotted, from the surface to the abyssal ocean.
- 'Lon-Vert' plots – (an uncommon option).

For all three: there may be multiple time-slices (i.e., you can plot data saved from different years).³

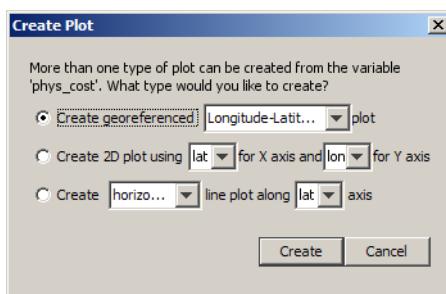


Figure 15.1: **Panoply** Create Plot dialogue window.

You can interpolate the data or not (often you may find that it is clearer not to interpolate the

²*WARNING* These instructions are strictly valid for older version of **Panoply** (ca. version 2.9.4), although some updates to the text have been made in light of version 4.6.2 ... so be aware that the plotting control buttons and options may have subtly changed in newer versions and the text no longer reflect the exact (current) options in **Panoply** ...

³Remember that the default, first time slice, will be the first once saved in the experiment. The last one saved and displayed, will reflect the end of your experiments.

data but to leave it as 'blocky' colors corresponding to the resolution of the model), change the scale and colors, overlay continental outline, change the projection, etc etc. Gray cells represent 'dry' grid points, i.e., continental or oceanic crust. To save plots in **Panoply** – from the file menu: File, then Save Image As ... and then select the location, filename, and graphics format.

15.2.1 Issues with Panoply default settings

The default settings in **Panoply**, i.e. those used when a plot is first created, can often mislead. In particular, note:

- Year (Arraytab)

The default is for the very 1st *time-slice* to be displayed rather than the experiment end. The first *time-slice* is numbered from 1 to however many total time-slices have been saved (displayed to the immediate right of the Year .box), and it is this integer number that appears in the Year box – not the year of the data save. Instead, the mid-point year of the time-slice is displayed in a second box (labeled 'Year mid-point').

Different *time-slices* to be plotted can be selected by either clicking through the saved year count, or by selecting the save year mid-point from the drop-down list.
- Scale Range (Scaletab)

The color scale is auto-scaled so that the range always goes from the minimum to maximum displayed value. This can potentially mislead if save years and/or depth/latitude slices are scrolled through as the scale will be automatically adjusted to fit each plot in turn.

Confusion can also arise for fields with no variation, e.g. atmospheric trace gas concentrations or air temperature – the auto-scaled plot in these instances has a uniform color but with odd hatching as Panoply dutifully tries to achieve the impossible (creating a scale of multiple colors for a single value).
- Zonal averaging (Array tab)

Lat-Vert plots are displayed as a zonal mean by default. This is indicated by the tick in the Ave box (bottom RH corner). Un-ticking the Ave box releases the averaging with the first longitudinal value of the grid now displayed instead. Similar to how Panoply displays years – the longitudinal grid locations are counter from 1 to typically 36 (depending on the resolution of the ocean grid), with the longitudinal mid-point value in degrees East displayed to the right.

Different longitudinal sections to be plotted can be selected by either clicking through the grid point number count, or by selecting the longitudinal mid-point from the drop-down list.
- Scale bar tick marks (Array tab)

The tick labels on the color scale are displayed by default in the format: x.y . If the typical values of the variable are order e.g. 10^{-6} you will end up with value labels ranging from 0.0 to 0.0 ... This can be most easily resolved in one of two ways:

 - The format of the label can be changed by selecting a different option from the pull-down Tick Label Format box (default == %.1f). For instance, %.2e would give a display in the format x.xxEyy (or x.xxE-yy) or %.6f would give x.xxxxxx.
 - An alternative is to re-scale the values. This is done in the Scaling Factor box in which you set the scale factor in powers of 10. For example: setting -6in effect converts units of mol kg⁻⁶ to μ mol kg⁻⁶.
- Other things to watch out for include:

- A plot involving depth, being by default, 'up-side-down'!.
This is fixed in the Gridtab, and the click button, Swap B/T.
- In 'Lon-Lat' plots, the modern continental outline being displayed by default.
This can be fixed by changing options in the Overlaytab.

Simply be careful when opening a new plot that you are looking at what you *think* you are looking at (or what you think you are looking at *is* what you are looking at).

Note that default plotting settings in **Panoply** can be changed (and saved).

15.2.2 Basic plots – examples

TO COME ...

15.2.3 Difference (anomaly) plots

It is possible to create an anomaly (difference) maps in **Panoply** which are essential when analyzing changes in a variable that may be small compared to the global spatial variability. To do this:

- First, open the netCDF results file.
- Open the variable of interest, e.g., atm_temp (surface air temperature) in the 2D netCDF file.
- From the upper LH corner of the Dataset Browser window, from the drop-down menu, select the name of the plot you have just created (atm_temp in field_biogem_2D...).
- From the upper LH corner of the Dataset Browser window, now click on the Combine Ploticon.

You now have a plot window that is displaying a difference map. By default, it is showing you the difference between two identical (in time) slices. The two different slices are labeled Array 1 (LH side) and Array 2 (RH side).

NOTE: Easier than mucking about with Combine Plot, is having open the first dataset, simply drag another variable from the list of variables in the Sources window, into the Plot window. You can drag either a different, or the same variable, into the Plot window.

For instance, you can keep one array (Array 1) fixed to the initial (year 1 (centered on 0.5)) and vary the year in the second array (Array 2). Note that you can select in Panoply whether Array 1 - Array 2 is plotted, or Array 2 - Array 1, or various proportional or relative differences.

If you switch off the auto-scaling feature (Always fit to data) you can center the scale so that no change is white, with positive deviations = red and negative = blue by clicking on Center on 0. This is something of a convention in the scientific literature.

The same variable in two different model experiments can also be opened up and analyzed combined:

- Start by opening up both required netCDF files.
- Open the variable of interest in one (either one) of the two 2D netCDF data-sets.
- From the upper LH corner of the Dataset Browser window, from the drop-down menu, select the name of the plot you have just created.

- Now double-click in the variable in the 2nd netCDF dataset.

You now have a plot window that is displaying a difference map, but of the same variable between two different experiments, rather than two years of the same experiment.

15.2.4 Ocean velocity plots

By combining the two (horizontal) fields of ocean circulation, rather than a difference plot, **Panoply** can create a velocity plot. This is a great way of visualizing surface (and deeper) currents and circulation patterns. To do this:

- First, open the 3-D netCDF results file.
- Open either the `phys_u` ('ocean velocity - u') or `phys_v` ('ocean velocity - v') field and select a Lon-Lat plot.
- From the upper LH corner of the Sources window, from the drop-down menu, select the name of the plot you have just created.
- Now double-click on the other velocity variable (whichever of the u and v fields you did not open first).
- By default you get a difference map, which is pretty useless really. From the drop-down Plot menu box (which should be displaying 'Array 1 - Array 2' by default) select: Vector Magnitude (bottom of the list).

You now have a plot window that is displaying the ocean velocity field, with arrows indicating the direction and speed (length of the arrow) together with an interpolated color background of the speed.

You can re-scale the velocity arrows to more clearly display the circulation pattern by altering the Scale Length value (Contours & Vectors tab). A value of 0.1 is a reasonable choice for surface currents. e.g. see Figure 15.2.

If you want to display deeper (in the ocean) current fields and/or different time-slices, take care that the depth level (/time-slice) in both LH and RH sides of the Array(s) panel must be changed to the same value. If displaying deeper current fields, then the velocity vectors will have to be further re-scaled (to a smaller value) in line with the lower velocities at depth compared to the surface.

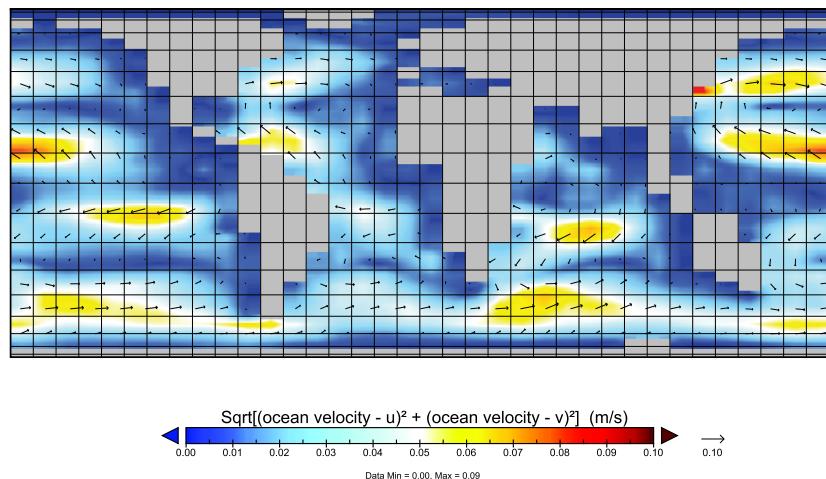


Figure 15.2: Example (modern) ocean surface velocity (current) map.

15.3 MATLAB plotting

15.3.1 MATLAB 101

If you need a tutorial on **MATLAB**, either as a refresher, or because you have not used the program before, refer to (and/or work through) the following sections of the [matlabananas MATLAB](#) textbook:

- Chapter 1 – this covers the very basics of using **MATLAB**, what variables are, including scalars (e.g. single numbers), vectors (1D array), matrices (2D array) and higher order arrays, how to index and carry out basic manipulation of arrays, basic data loading and saving and plotting.
- Sections 2.1 and 2.2 of Chapter 2, which cover creating (script) programs – i.e. adding all your **MATLAB** commands to a file (a .m or 'm-file') and running the file, and *functions* – programs (m-files) where one or more parameters might be passed into the *function* when it is called (e.g. at the command line), and potentially variables returned.
- In Chapter 3 – subsection 3.1.3 deals with reading in netCDF format data, which is the **muffin** format for spatial data. Section 3.2 deals with more advanced 2D plotting (also of direct relevance to **muffin** data processing and visualization using **MATLAB**).

15.3.2 MATLAB and ASCII (*time-series*)

The *time-series* (.res) output of **BIOGEM** are in a simple plain text (ASCII) format. These can be read in very easily in **MATLAB** using the `load` command. Note that a brief description of each column of data in the *time-series* files appears on the first line of the file, and that prefixing this is a % symbol, that **MATLAB** ignores. Hence only the columns of data gets read in by default using `load :)`⁴ For example:

```
>> co2=load('biogem_series_atm_pCO2.res','ascii');
>> plot(co2(:,1),1.0E6*co2(:,3));
```

loads in the atmospheric CO_2 time-series file (assigning it to the array variable `co2`), and then plots (as a line graph) the 3rd column (CO_2 concentration) vs. the first (time). Because concentrations are saved in units of atmospheres, a factor of 1.0E6 is applied to convert to μatm . Equivalently:

```
>> co2=load('biogem_series_atm_pCO2\).res','ascii');
>> scatter(co2(:,1),1.0E6*co2(:,3));
```

15.3.3 MATLAB 'Import Data' ...

You can also import the time-series files by clicking on the Import Data icon:

- Navigate to the **BIOGEM** sub-directory of your experiment results directory. Ensure that All Files is selected and click on the time-series file you want.
- **MATLAB** ignores the header lines, and it should be safe to simply click on Import Selection for all columns, or select what you want to plot – typically year (the first column) and another one.
- A default variable name – the filename minus the underscore characters – will appear listed in the **MATLAB** Workspace window. Double-click on the variable name to open up the

⁴There are also other **MATLAB** commands for reading in text data – refer to the [MATLAB programming text](#).

imported data in a table view. If you select columns to plot, and then head over to the main PLOTS tab, a range of plotting options are provided and a plot can be generated by clicking on one of the plotting icons.⁵

- Labels can be added, scales and markers changed, etc etc, in the Figure window .

Note that as the data has been imported as an array in **MATLAB**, you can also plot directly from the command line. (And indeed, you could also have loaded the data from the command line.)

15.3.4 MATLAB and netCDF (*time-slice*)

Basically – the only hard part, having opened the *netCDF* file is in correctly deducing which dimension in an extracted data array is longitude, latitude, (sometimes depth), or time. Mostly this should be pretty obvious from inspecting the **MATLAB** Workspace window (assuming you have a column for Size selected to be displayed), or using the `size` command.⁶ Once you have done this, you can plot slices, scale data, average or otherwise process data, extract locations at specific locations, etc etc.

As an example – in **MATLAB**, first either change directory to the biogem directory of a set of **muffin** results, or from where-ever you are, add a path to the location of the biogem directory. To open the 2D **BIOGEM** netCDF results file, type:

```
>> ncid = netcdf.open('fields_biogem_2d.nc','nowrite');
```

To extract a variable, you first need to find its ID from its name:

```
>> varid = netcdf.inqVarID(ncid,NAME);
```

where NAME is a place-holder for the name of the variable (as a string). You might need to use **Panoply** to display all the different variable names. Or, you can list all the variables and stuff in the netCDF file using `ncdisp`:

```
>> ncdisp('fields_biogem_2d.nc')
```

Having, by one means or another, identified the name of the variable you are interested in, you can recover its ID and then the data itself, for example, for `atm_temp` (surface air temperature):

```
>> varid = netcdf.inqVarID(ncid,'atm_temp');
>> data = netcdf.getVar(ncid,varid);
```

In loading in the variable, you end up with a multi-dimensional array – 2 spatial dimensions and if you have more than 1 *time-slice* of data saved, 1 temporal dimension (and if you loaded in the **BIOGEM** 3D netCDF file, you end up with a 4-dimensional array). **MATLAB** reports the size of the array in the Workspace window (depending on which column display options you have selected). In the example here, which took the experiment EXAMPLE.worjh2.Caoetal2009.RCP6p0, the array for atmospheric temperature is reported as $36 \times 36 \times 8$.⁷ so the last of the 8 time-slices would be accessed as:

⁵Note that if you have not selected any data columns, then all the plotting icons are disabled and greyed out in the PLOTS tab.

⁶It also turns out that the order of dimensions for a variable read in by **MATLAB**, is the opposite of the order listed in the Variable window of **Panoply**.

⁷Although note that in **Panoply**, it is reported as `atm_temp(time=8, lat=36, lon=36)`.

```
>> data_last = data(:,:,8);
```

or

```
>> data_last = data(:,:,:,end);
```

Panoply reports variables in the the **BIOGEM** 3D netCDF file as having dimensions of: (*time*, *zt*, *lat*, *lon*) (e.g. (time=13, zt=16, lat=36, lon=36)) whereas **MATLAB** reads it in the reverse order. For instance, if we load in the (3D) ocean temperature field:

```
>> ncid = netcdf.open('fields_biogem_3d.nc','nowrite')
>> varid = netcdf.inqVarID(ncid,'ocn_temp');
>> data = netcdf.getVar(ncid,varid);
```

and then type:

```
>> size(data)
ans =
    36     36     16     13
```

we get an array orientated as (*lon*, *lat*, *zt*, *time*).⁸ The final time-slice would hence then be accessed:

```
>> data_last = data(:,:,:,end);
```

with *data_last* now becoming a 3D array with dimensions (*lon*, *lat*, *zt*).

There are three key things to remember at this point:

1. Firstly, the depth levels are read such that index 1 is the surface, and 16 is the deepest ocean depth level (in this case, otherwise it is 8).
2. For the lon-lat part – (*lon*, *lat*) equates to *rows* vs. *columns* in **MATLAB**, and hence if you were to plot e.g. the surface ocean slice:

```
>> imagesc(data_last(:,:,1));
```

you will end up with the plot on its side, with latitude along the *x*-axis and longitude on the *y*-axis.

You can exchange rows and columns in **MATLAB** with the *transpose operator* (see programming/**MATLAB** book):

```
>> imagesc(data_last(:,:,1)');
```

but ... this still leaves you with an up-side-down plot, because **MATLAB** reads from the first row down, whereas in latitude, you are expecting to read from -90 degrees up (towards the N pole). *flipud* accomplishes the final transformation:

```
>> imagesc(flipud(data_last(:,:,1)'));
```

⁸*zt* is the depth level in the ocean.

3. The final complication is that **muffin** netCDF output uses a special value to represent an invalid or null number, e.g. for ocean temperature where the grid point is land (and an ocean temperature value would have no meaning). In the netCDF definition, **Panoply** is told what the special number is, hence it shows land in grey when plotting ocean variables.

Panoply reports this as: `missing_value = 9.969209968386869E36`.

There is no easy way (that I can see!) to get **MATLAB** to deal with this for you, so you need to search and replace this value, e.g.:

```
>> null=9.969209968386869E36;
>> data_last(find(data_last==null))=NaN;
```

which searches for this null value and replaces it with a NaN (so that now it can be simply plotted).

Once you are done accessing data, it is good practice to close the netCDF file after you are done with it:

```
>> netcdf.close(ncid);
```

The question then remains: what do you actually 'do' with it in **MATLAB**?

Plotting sections

For a quick look-see, `imagesc` (as per above) is handy.

For more advanced plotting (and for presentation) – refer to the **matlabananas MATLAB** programming text.

As per above, horizontal (lon-lat) fields can be extracted from 2D *netCDF* output by: `data(:, :, time)` and from 3D by: `data(:, :, zt, time)`. Obviously, you could also extract lon-depth via: `data(:, lat, :, time)` and lat-depth via: `data(lon, :, :, time)`.

Anomalies (with time) are created: `data(:, :, :, time2)-data(:, :, :, time1)`.

Typically, except a little trial-and-error in extracting the dimensions you want, and also in correcting the orientation of the matrix or resulting plot.

Calculating inventories

Often, the inventory (total mass or number of moles) of the ocean or atmosphere is useful to know, particularly as a function of time. For the ocean (or atmosphere) as a whole, the *time-series* output files report this (alongside the mean global concentration).

You can also calculate this with **MATLAB** from the netCDF output. Fields of ocean concentration are saved in the 3D output (and atmospheric concentrations in the 2D output). To convert concentration in the ocean, in units of $mol kg^{-1}$ you'll need to know the mass of each ocean cell, in units of kg .

When saving using save option 9, or 99^9 , you save the 'physics' of the ocean model, which is actually mostly just the grid information, such as cell area, thickness, latitude and longitude edges and midpoints, depth edges and mid point. Also saved are the masses and volumes of the grid of cells. So to derive an array of cell tracer inventories from an array of concentrations, and the array of cell masses (variable `phys_ocn_M`), you'd write:

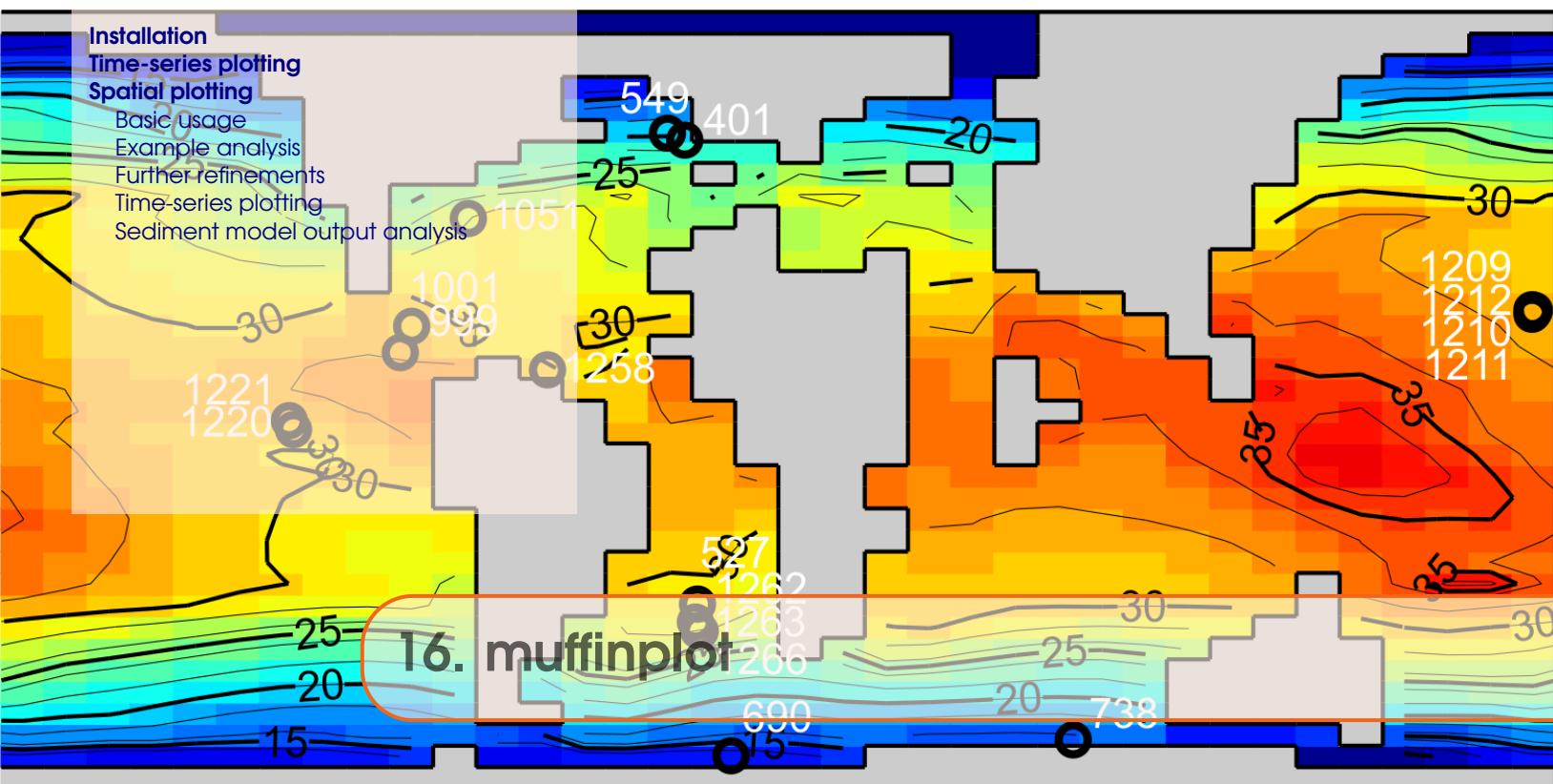
⁹Parameter `bg_par_data_save_level` – see earlier.

```
varid = netcdf.inqVarID(ncid,'ocn_temp');
data  = netcdf.getVar(ncid,varid);
varid = netcdf.inqVarID(ncid,'phys_ocn_M');
mass  = netcdf.getVar(ncid,varid);
inventory = data(:,:,:,:time).*mass(:,:,:,:time);
```

Before summing inventory to determine the global total inventory, you will, as before, have to deal with the null values (converting them to NaN) and then deal with the presence of NaNs in the array when summing ...

The advantage of doing the calculations in **MATLAB** (despite being provided with the global mean and inventory in the time-series files) is that you could calculate the inventory of a tracer (/substance) for just the ocean surface, or just a specific region of band of latitude. Or you could calculate the mean concentration for just a specific region of the ocean.¹⁰

¹⁰In calculating mean concentrations, you'll need to volume or mass weight the concentrations, and hence still need to use one of the physics variables.



The **muffinplot** suite of **MATLAB** functions provides a means of plotting a variety of output reproducibly (by means of saved parameter file) and with the potential for automation (i.e. automatically generating the same analysis for a large number of different experiments).

The functions comprising this software suite include:

- `plot_fields_biogem_2d`– lon-lat plots from the 2D **biogem** output.
- `plot_fields_biogem_3d_i`– lat-depth plots from the 3D **biogem** output.
- `plot_fields_biogem_3d_k`– lon-lat plots from the 3D **biogem** output.
- `plot_fields_ccd`– analysis of the 'CCD' (from both **biogem** 2D and **sedgem** 2D output).
- `plot_fields_sedgem_2d`– lon-lat plots from the 2D **sedgem** output.
- `plot_histc_2d`– a generic color-coded histogram function.
- `plot_sedcore`– down-core plots from **sedgem** sedcore output.
- `plot_timeseries_biogem`– time-series plots from **biogem** time-series output.

Note that at this current time, there is no facility for lon-depth plotting.

Most of the plots also perform additional functions (which can be generally disabled if not wanted), such as plotting and saving zonal or depth profiles, plotting difference maps, plotting and labelling data on maps and carrying out model-data fit statistics and plotting, extracting model values at data locations.

The following sections provide an overview and examples of such plotting and analysis.

[Installation](#)
[Time-series plotting](#)
Spatial plotting
[Basic usage](#)
[Example analysis](#)
[Further refinements](#)
[Time-series plotting](#)
[Sediment model output analysis](#)

16.1 Installation

muffinplot can be obtained from [github](#). If you do not have a **git** client on your computer (and hence can clone the repository locally), then simply download an archive of the code (from [clone](#) or [download](#) – pick Download ZIP).

When you unpack (or clone) **muffinplot**, you should see 3 directories – EXAMPLES, and MASKS, source, a series of .m files, and a single lonely .ps graphics file (colorscales.ps). The .m files are split into filenames with or without the label SETTINGS – the ones without are code files (*functions*), and the ones with the word SETTINGS in their filename, contain parameter settings for plotting.

By default (the parameters can be changed if you wish), wherever you run the **muffinplot** plotting function from, requires that you have a subdirectory called cgenie_output, where you will place the (complete) **muffin** experiment output directories (i.e. the contents of cgenie_output should look like the contents of cgenie_output on your cluster account¹). If you do any model-data analysis, by default, a directory DATA is also expected. Mask files reside in the MASKS subdirectory of the **muffinplot** installation, or in your current directory, or anywhere in the **MATLAB** path.

The simplest option might be to unpack/clone **muffinplot** to a directory containing cgenie_output and hence all your experimental results directories.² In other words:

1. Create some local directory e.g. called RESULTS.
2. Create a subdirectory in RESULTS called cgenie_output.
3. Drag your experiment results folders across into the subdirectory cgenie_output, as per the directory structure on the cluster. (Or drag the archived files and unpack them.)
4. Install **muffinplot** in the RESULTS directory.
5. Change the **MATLAB** working directory to RESULTS.

The plotting functions are run simply by typing their name and passing a list of parameters (comma-separated, with the complete list enclosed in parentheses). By default, the SETTINGS files need to be in the same directory as you are running the *functions* from, or in one of the **MATLAB** paths. Results are saved to a subdirectory that by default is called PLOTS, which will be created for you if it does not already exist.

All the plotting functions provide some manner of 'help', that can be obtained by typing at the command line:

```
>> help FUNCTIONNAME
```

where FUNCTIONNAME is the function name (as per listed above).

16.2 Time-series plotting

The **muffinplot** function plot_timeseries_biogem.m provides a facility to plot **BIOGEM time-series** (.res) output.³ You can use **MATLAB** help on the function name to detail the parameters that need to be passed (and examples).

¹Although you do not need to copy all the results over ... just the experiments that you wish to plot up.

²However, the **muffinplot** functions do not have to be run from the same directory that you are in – you can install them somewhere convenient, and then with **MATLAB** set to a directory containing a cgenie_output experiment results directory, you can:

```
> addpath(PATH)
```

where PATH is the path to the directory where **muffinplot** is installed.

³Obviously – there are lots of different and easy ways of plotting plain text output in the form of a simple column format.

The `plot_timeseries_biogem` plotting function plots a basic set of time-series variables by default. It then, enables a set up up to 3 additional variables to be plotted. It is also associated with a file of parameter values (`plot_timeseries_SETTINGS.m` by default) for fine-tuning plots.

The plotting function requires a list of parameters to be passed in the argument list, i.e.:

```
>> plot_timeseries_biogem(PAR1,PAR2,PAR3, ... PARn)
```

These are, in order:

1. PEXP1 – *string* → the (first) experiment name.
2. PEXP2 – *string* → is the name of the 2nd (optional) experiment. If no second experiment is selected, then a null string value must be passed, i.e., ''.
3. PTMIN – *real* → minimum plotted time (x-axis).
4. PTMAX – *real* → maximum plotted time (x-axis).
5. PDATA1 – *string* → time-series variable name for additional data to plot.
Omit the '`biogem_series_`' and '`.res`' parts of the filename.
Leave blank, i.e., '', for no additional data panel.
6. PDATA1N – *integer* → the column number of the data in the time-series file.
7. PDATA2 – *string* → time-series variable name for additional data to plot.
8. PDATA2N – *integer* → the column number of the data in the time-series file.
9. PDATA3 – *string* → time-series variable name for additional data to plot.
10. PDATA3N – *integer* → the column number of the data in the time-series file.
11. POPT – *string* → The string for an alternative plotting parameter set.
If an empty ('') value is passed as this parameter, then the default parameter set file is used.
12. PNAME – *string* → The string for an alternative filename.

Note that if an empty value is passed as this parameter, then a filename is automatically generated.

A simple example usage would be:

```
>> plot_timeseries_biogem('myexperiment','','0.0,10000.0','','0','','0','','0','','')
```

where `myexperiment` is the name of the 1st experiment, followed by an empty string ('') indicating no second experiment. The results are to be plotted from 0.0 to 10000.0 years (the 2 following parameters; 0.0, 10000.0). Then, no additional (maximum 3) optional parameters are requested, and hence the next parameters passed are: '', 0, '', 0, '', 0. Finally, the default plotting parameter set is required, and no specific alternative filename is to be used, accounting for the final 2 empty strings passed.

By default, `plot_timeseries_biogem` plots 2 panels of data, both with 2 (LH and RH) axes:

1. Atmospheric CO_2 . Note that if the experiment was not CO_2 -enabled (i.e. not run with a global carbon cycle), a warning is given and 'fake' data (actually, random numbers) is plotted.
2. Atmospheric $\delta^{13}CO_2$. Note that if the experiment was not $\delta^{13}CO_2$ -enabled (i.e. not run with a global carbon cycle), a warning is given and 'fake' data is plotted.
3. Atmospheric temperature (as a global mean, annual average).
4. Fractional (percentage) sea-ice extent.

Additional model outputs can then be added by listing them in the function call. For example, to also plot the global overturning strength, which is contained in the file `biogem_series_misc_opsi.res`, you would add '`misc_opsi`', 3⁴, where the 3 indicates the 3rd column of data in the file is to be

⁴Don't forget that you omit the '`biogem_series_`' and '`.res`' parts of the filename.

plotted, which in this case is the global maximum overturning value (and the 2nd column is the minimum value). The complete line looks like:

```
>> plot_timeseries_biogem('myexperiment','','0.0,10000.0,'misc_opsi',3,'',0,'',0,'',')
```

By default, the plotted variables are all auto-scaled. To specify the y-axes, you will need to edit the plotting settings parameter file: `plot_timeseries_SETTINGS.m`. For the default plotted 4 results variables, the minimum and maximum y-axis limits are specified in the section:

```
axis_pCO2min = 0.0;
axis_pCO2max = 0.0;
axis_d13Cmin = 0.0;
axis_d13Cmax = 0.0;
axis_Tatmmin = 0.0;
axis_Tatmmax = 0.0;
axis_icemin = 0.0;
axis_icemax = 0.0;
```

The default zero values here, tell the plotting function to create an auto-scale for the y-axis.⁵ Following this in the parameter file, are the settings for the optional variable plotting:

```
axis_data1_min = 0.0;
axis_data1_max = 0.0;
axis_data2_min = 0.0;
axis_data2_max = 0.0;
axis_data3_min = 0.0;
axis_data3_max = 0.0;
```

Note that if you want instead to copy and rename and then edit this settings file, you will need to pass the new (non-default) filename when calling the plotting function. For example, if you created a new parameter settings file: `settings_NEW.m`, then the *function* call would look like:

```
>> plot_timeseries_biogem('myexperiment','','0.0,10000.0','','0','','0','','0,'settings_NEW','')
```

16.3 Spatial plotting

Overview

4 of the **muffinplot** plotting functions provide spatial (2D) plotting capabilities:

- `plot_fields_biogem_2d`
Plot a 2-D field from: `fields_biogem_2d.nc`.
- `plot_fields_biogem_3d_i`

Plot a vertical-meridional (2-D) slice through the ocean (i.e., all cells have the same *i* (longitudinal) coordinate value) from: `fields_biogem_3d.nc`.

Options are provided for averaging longitudinally over a supplied mask, which may be the entire ocean and hence giving a global meridional cross-sectional mean, of a specific ocean basin, or may be a single cell 'wide' longitudinally and take a meandering path hence simulating an ocean transect. An option is also provided to overlay an ocean circulation stream-function.

⁵Note the units of atmospheric CO_2 as μatm .

- `plot_fields_biogem_3d_k.m`

Plot a horizontal slice through the ocean from: `fields_biogem_3d.nc`.

An option is provided for overlaying ocean circulation (velocity fields). Water column integrals can also be calculated and displayed, as well as benthic surfaces, and the function can also determine the spatial distribution of the maximum or minimum value occurring anywhere in the water column (or portion of the water column).

- `plot_fields_sedgem_2d`

Plot a 2-D field from: `fields_sedgem_2d.nc`.

All 4 plotting functions can also overlay observed data and create difference (anomaly) maps – either between different experiments, time-slices, or variables, or between model and data and provide summary statistics regarding the difference.

Argument (parameter) list

All 4 plotting functions share exactly the same format of parameters⁶ passed in the argument list:

```
>> FUNCTIONNAME(PAR1,PAR2,PAR3, ... PARn)
```

i.e. take a (long!) list of parameters. These are (in order):

Firstly, a series of parameters for defining experiment, variable, and year:

1. PEXP1 – *string* – is the name of the 1st (main) experiment. A results directory with the same name must exist in the directory `cgenie_output`⁷.
2. PEXP2 – *string* – is the name of the 2nd (optional) experiment. If no second experiment is selected, then a null string value must be passed, i.e., ''.
3. PVAR1 – *string* – is the name of the 1st (main) variable. If no valid variable value is given, a list of valid variable names will be printed out.⁸
4. PVAR2 – *string* – is the name of the 2nd (optional) variable. If no second variable is selected, then a null string value must be passed, i.e., ''.
5. PT1 – *real* (or *integer*) – is the value of the 1st (main) time-slice. If no valid variable value is given, a list of valid variable names will be printed out.⁹
6. PT2 – *real* (or *integer*) – is the value of the 2nd (optional) time-slice. If no second time-slice is selected, then enter -1.¹⁰

Then there are 2 parameters for plotting sub-sets of the 2D or 3D data (essential for 3D data which cannot be usefully visualized in raw form):

1. PIK – *integer* – varies in its interpretation and is discussed below.
2. PMASK – *string* – is the name of an optional (2D) mask. A null string ('') must be passed if no mask is requested. A file with the same name (plus an extension .dat) must exist in the directory `MASKS`¹¹. The interpretation of this parameter differs slightly between functions (below).

Next come options for plotting scale control:

⁶Parameters can be in for form of strings, in which case they must be given as a series of characters enclosed in inverted commas ''; as real numbers, e.g. 999.5 or 9.995E2; or integers, e.g. 2, 10.

⁷Or alternative directory if the default file path settings have been changed.

⁸As a string, the value must be encased in inverted commas: ''.

⁹As `sedgem` does not save multiple and/or time-specific data, a dummy value (anything) is entered here.

¹⁰As `sedgem` does not save multiple and/or time-specific data, a dummy value (anything) is entered here.

¹¹Or alternative directory if the default file path settings have been changed.

1. PCSCALE – *real* (or *integer*) – is the scale factor for the plot. For example, to plot in micro molar (umol kg⁻¹) units, enter; 1e-6. The plot is auto-scaled if a value of zero (0.0) is entered.
2. PCMIN – *real* (or *integer*) – is the minimum scale value.
3. PCMAX – *real* (or *integer*) – is the maximum scale value.
4. PCN – *integer* – is the number of (contour) intervals between minimum and maximum scale values.

Finally, there are 3 parameters for: specifying discrete (observed) data to be plotted (and analyzed against model projections), for specifying the plotting parameter file to be used, and for substituting an alternative filename for all the output:

1. PDATA – *string* – is the filename containing the an overlay data set, which must be formatted as separated columns. The precise number and type of columns varies between different functions and also the plotting options chosen, and are hence discussed later. The full filename of this file must be give, including any extensions (e.g. .dat, .txt). This parameter must be passed as a *string*; leave blank, i.e., '', for no overlay data.
2. POPT – *string* – is the m-file filename (excluding the .m extension) containing the plotting options (SETTINGS). This parameter must be passed as a string; leave blank, i.e., '', in order to load the default file (plot_fields_SETTINGS).
3. PNAME – *string* – is the string for an alternative series of output filenames. This parameter must be passed as a string, e.g., 'experiment2'. If an empty (i.e., '') value is passed to this parameter then the output filenames will be automatically generated.

The basic parameter list for all 4 plotting functions¹² is hence:

```
>> FUNCTIONNAME(PEXP1,PEXP2,PVAR1,PVAR2,PT1,PT2,PIK,PMASK,PCSCALE,PCMIN,PCMAX,PCN,PDATA,POPT,PNAME);
```

Function specific interpretation of PIK and PMASK

A note on the different behaviour of 2 of the passed parameters, depending on which plotting function is used – PIK, and to some extent, PMASK, have quite different interpretations depending on the particular plotting function used:

1. plot_fields_biogem_2d
 - (a) PIK – is the maximum depth (k) level that will be plotted, i.e. all depth levels deeper than PIK will be excluded. This is useful for plotting a variable only for the 'deep' ocean (rather than the ocean overlaying all ocean depths) for example. This value also provides an alternative way of creating a mask, and only values of k less than or equal to the passed value will be plotted.
 - (b) PMASK – is the name of an optional (2D) mask. A null string ('') must be passed if no mask is requested. (Shallow depths could also be excluded from the plot by means of a mask rather than setting PIK.)
2. plot_fields_biogem_3d_k
 - (a) PIK – the depth (k) level to be plotted. Note that the levels are numbered from a maximum value designating the surface, to 1 for the deepest ocean level. Typically, maximum values for the number of ocean levels are 8 (e.g. Ridgwell *et al.* [2007]) or 16

¹²Note that for plot_fields_sedgem_2d several of the parameters are redundant but must still be included (typically as zeros). This is in order to retain a common parameter list format between all the different plotting functions.

(e.g. *Cao et al.* [2009]).

Non ocean level k values have special meanings here:

- i. 0

A zero will result in a water column integral being plotted. With data, the model-data is carried out on the grid as a whole.

- ii. -1

Will result in the benthic surface being plotted.

- (b) MASK – is the name of an optional (2D) mask. A null string (‘’’) must be passed if no mask is requested.

3. plot_fields_biogem_3d_i

- (a) PIK – the longitude-depth (i) slice through the ocean to be plotted.

Non longitude grid point i values have special meanings here:

- i. 0

A zero will result in a zonal mean being plotted. With data, model-data comparison is conducted at the specific data locations, rather than vs. a zonal mean model value.

- ii. -1

I have forgotten what this does ...

- (b) MASK – is the name of an optional (2D) mask. A null string (‘’’) must be passed if no mask is requested.

For example: if the mask is of the entire ocean (`mask_worbe2_ALL.dat`), the result is a global meridional cross-sectional mean.

If the mask is just of a single basin such as the Atlantic (`mask_worjh2_Atlantic.dat`), the result is the Atlantic meridional cross-sectional mean.

Masks can also be constructed that are only a single cell wide longitudinally, but which take a meandering path following an ocean transect¹³.

The trivial usage would be to construct a mask consisting of a vertical line of 1s – the result is equivalent to setting an appropriate i value in PIK.

4. plot_fields_sedgem_2d.m

is an exception as it does not (currently) use either parameter.
PIK must be entered as 0 (any integer will do in fact), and PMASK as ‘’’.

The mask itself (if PMASK contains a mask name) is a 2-D array of model grid points (on the **BIOGEM** grid) in the form of a simple ASCII file. A value of '1' represents a vertical column of ocean cells to include, whereas a value '0' will exclude all cells in the water column at that particular grid point. Examples of some masks can be found in the MASKS subdirectory of **muffinplot**.

¹³e.g., as in: `mask_worjh2_GEOSECS_WATL.dat`

16.3.1 Basic usage

What follows are some basic and quasi random examples, just to illustrate a simple use of the three main plotting functions.

1. Surface ocean temperature

Surface ocean temperature can be plotted in 2 ways – via the 2d plotting function (but only if the surface tracer properties fields have been saved, as these are optional), or via the 3d plotting function.

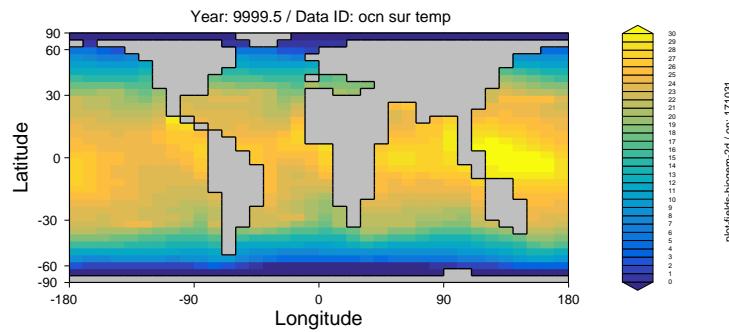


Figure 16.1: Example basic (default) surface temperature plot.

For example:

```
>> plot_fields_biogem_2d ...
('EXP1', '', 'ocn_sur_temp', '', 9999.5, -1, 16, '', 1.0, 0.0, 30.0, 30, '', 'example1a');
```

plots from the experiment EXP1, the variable `ocn_sur_temp` for time-slice 9999.5 (the mid-point time of the final year of a 10,000 year experiment). The color scale is from 0.0 to 30.0, with no re-scaling (1.0), and 30 color intervals in the scale. The default SETTINGS parameter file is used, and the default filename string replaced with `example1a`. The only other thing to note, is for parameter PIK, a value of 16 is set – corresponding to the ocean surface. See Figure 16.1.

Note that in this example, the variable `ocn_sur_temp` is assumed. However, the model results variable `ocn_sur_temp` is not always saved in **BIOGEM** 2D netCDF output. If the requested variable, such as `ocn_sur_temp` does not exist, or is mis-spelt, **MATLAB** will pause and provide a warning. It will then list all the variables in the netCDF file that it can find and wait for a new variable name to be inputted. For instance, a variable that is always saved in **BIOGEM** 2D netCDF output is `atm_temp` (surface air temperature) and could be substituted in the plot. Also note that time (the time-slice year to be plotted) is similarly handled – if the specific time-slice value does not exist, a list of all possible time-slice years are provided and a substitute value requested as **MATLAB** pauses and wait for your input.

To add contours, in the default SETTINGS parameter file (or a copied and re-named version thereof), adjust the following line:

```
contour_plot = 'y'; % [ 'y' ] OVERLAY CONTOUR PLOT?
```

The results of this are shown in Figure 16.2.

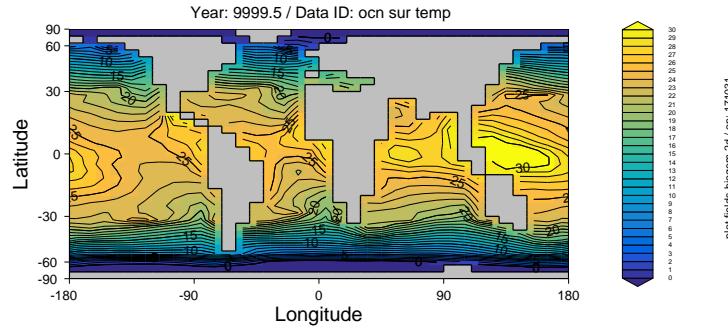


Figure 16.2: Example surface temperature plot, with contours.

Refinements to the contouring can be done by changing the lines:

```
contour_mod = 1; % [ 1] NUMBER OF COLOR INTERVALS PER CONTOUR
contour_mod_label = 5; % [ 5] NUMBER OF LABELED CONTOURS PER CONTOUR
contour_label = 'y'; % [ 'y'] LABEL CONTOURS?
contour_dashneg = 'n'; % [ 'n'] PLOT NEGATIVE CONTOURS DASHED?
```

(these are the more commonly used refinements).

Here: `contour_mod` determines how many color intervals per contour interval, which in the previous SST plot example, was 30. So a value of 1 will give you 30 contours – one every 1 degree C. And a value of 5 will give you 6 contours – one each 5 degrees C.

`contour_mod_label` then determines whether you want the contours labelled or not. The answer (parameter value) to be the character `y` or `n`, as a string (i.e. in inverted commas). If you elect to have contour labels, `contour_mod_label` determines how frequently to label the contours. A value of 1 labels every single contour. A value of 2 labels every other contour. So for instance, if you set `contour_mod=5` and `contour_mod_label=2` in the previous SST example, you get a contour every 5 degrees C, and a temperature label every 10 degrees C.

Alternatively, using 3d plotting, you could plot the ocean surface temperature field as follows:

```
>> plot_fields_biogem_3d_k ...
('EXP1', '', 'ocn_temp', '', 9999.5, -1, 16, '', 1.0, 0.0, 30.0, 30, '', '', 'example1c');
```

The main things that change here are firstly the variable name – now `ocn_temp`, and secondly because this is a 3D ocean field, we need to specify what ocean model level we want to plot – this is where the integer 16 comes in and corresponds to the input PIK, as discussed earlier. The resulting plot will be identical to Figure 16.1.

2. Global zonal average temperature profile

To keep with ocean temperature, we can use the `plot_fields_biogem_3d_i` function to plot the global zonal mean (lat-depth) profile (rather than horizontal, surface slice):

```
>> plot_fields_biogem_3d_i ...
('EXP1', '', 'ocn_temp', '', 9999.5, -1, 0, '', 1.0, 0.0, 30.0, 30, '', '', 'example2a');
```

The only significant change as compared to before, is setting a 0 for input parameter PIK (again – see earlier). The results is shown in Figure 16.3.

3. Pacific dissolved oxygen profile

As per choosing ocean levels (`k`-values) in the lon-lat plotting, you can also specify a specific longitude for creating a lat-depth section rather than calculating and plotting a global zonal

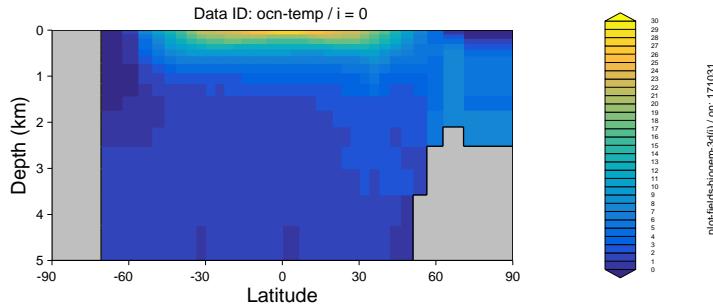


Figure 16.3: Zonal mean global ocean temperature profile.

mean. e.g. Figure 16.4 was created by¹⁴:

```
>> plot_fields_biogem_3d_i ...
('EXP1', '', 'ocn_02', '', 9999.5, -1, 10, '', 1.0E-6, 0.0, 300.0, 30, '', '', 'example3a');
```

The chosen section is somewhere in the Pacific, along a line of longitude (whatever corresponds to $i=10$ on this **muffin** model grid ... I guess about 165W ...). Here, the variable to be plotted has also been changed – `ocn_02`¹⁵. Because the units of dissolved oxygen are much smaller than for temperature (in degrees C), the plotted scale has also been changed – from 0 to 300 $\mu\text{mol kg}^{-1}$ rather than the netCDF variable units of mol kg^{-1} . To achieve this re-scaling, a units scaling value of 1.0E-6 is specified for parameter PCSCALE.¹⁶

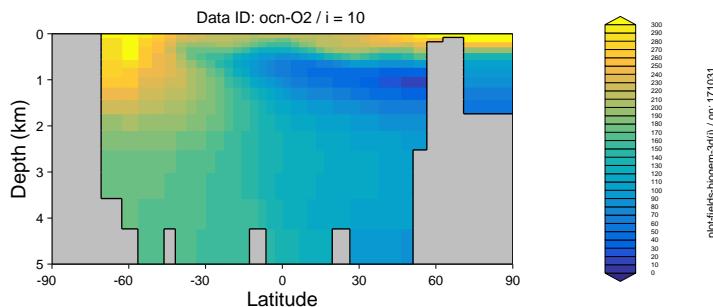


Figure 16.4: Ocean oxygen profile on a Pacific transect.

4. Atlantic zonal mean dissolved oxygen profile

So far, with the exception of plotting a gridded color field and a contoured field at the same time, all these examples can also be done in **Panoply**. One difference, is the ability in the **muffinplot** suite of **MATLAB** functions to apply masks – isolating geographical regions or even single points. In the MASKS directory, are a series of example ASCII mask files, mostly for the 2 (8- and 16-level ocean) modern published configurations of **muffin**. For instance, `mask_worjh2_AtlanticALL.dat` has all the grid points in the entire Atlantic basin assigned a value of 1, with 0 everywhere else. If we apply this first to the surface ocean dissolved oxygen field:

¹⁴Also turning on the contour plotting.

¹⁵You'll need a biogeochemistry enabled *base-config*

¹⁶Note that the scaling specified is as the new units relative to the old units – here, $\mu\text{mol kg}^{-1}$ relative to mol kg^{-1} and hence 10^{-6} . ALSO NOTE that **Panoply** does it the other way around ... :(

```
>> plot_fields_biogem_3d_k ...
('EXP1','','ocn_02','','9999.5,-1,16,'mask_worjh2_AtlanticALL.dat',1.0E-6,0.0,300.0,30,
... '','',,'example4a');
```

we obtain Figure 16.5.

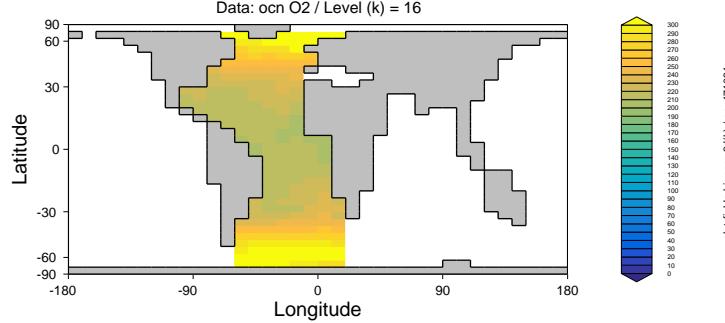


Figure 16.5: Distribution of surface ocean dissolved oxygen in the Atlantic.

Here, it is clear how the masked has been applied and all the ocean falling outside of the mask is plotted as white (no data).

We can also apply a mask field to the zonal average plot:

```
>> plot_fields_biogem_3d_i ...
('EXP1','','ocn_02','','9999.5,-1,0,'mask_worjh2_AtlanticALL.dat',1.0E-6,0.0,300.0,30,
... '','',,'example4b');
```

(Figure 16.6)

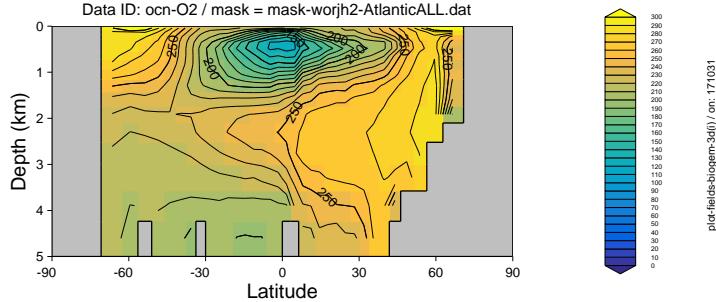


Figure 16.6: Mean zonal ocean oxygen profile in the Atlantic.

16.3.2 Example analysis

An set of **MATLAB** m-file functions are provided that define a series of different generic and basic experiment analysis and plottings, with `make_analysis_ALL.m` provided as a template for carrying them all out in one go. Obviously, the individual aggregate plotting functions can be edited, added to, or with unwanted or irrelevant plots, commented out or deleted – treat these all simply as templates for developing your own analysis strategy (as well as viewing the associated configuration files as illustrations of the function/use of some of the different further plotting options¹⁷).

The aggregate plotting functions are as follows:

¹⁷Also covered in a subsequent sub-sub-section.

1. **`fun_make_analysis_phys.m`**

This encompasses a basic set of analyses of ocean circulation and climatology.

The script is written as a *function*, and requires just two parameters to be passed as input:

- (a) The experiment name.
- (b) The (mid-point of the) year of the time-slice to plot.

In the example of an experiment called 'EXP1', and plotting the last annual time-slice (9999.5) from a 10,000 year model run, the function is hence called:

```
fun_make_analysis_phys('EXP1', 9999.5);
```

2. **`fun_make_analysis_geo.m`**

This encompasses a basic set of analyses of ocean (abiotic) geochemistry and ocean acidification related variables and metrics.

3. **`fun_make_analysis_bio.m`**

This encompasses a basic set of analyses of marine biological fluxes and biologically related properties.

4. **`fun_make_analysis_ALL.m`**

Aggregates all the above functions (i.e., calls all 3).

To run these example analysis – either copy all the files contained in the EXAMPLES subdirectory, to your working directory (e.g. RESULTS as per the previous example). Then type e.g..¹⁸

```
fun_make_analysis_ALL('EXP1', 9999.5);
```

OR, add the path to the EXAMPLES subdirectory, e.g.

```
addpath('Y:\_git\muffinplot\EXAMPLES');
```

but obviously depending on quite where you installed **muffinplot**.

16.3.3 Further refinements

A number of additional options for exerting finer control over the plotting are provided as a block of parameters and (default) values in the m-file itself, in a section immediately after the commented help and change-log at the start of the m-file. Not all the options are relevant to all the plotting functions¹⁹, but the full list (and then defaults in brackets []) is as follows:

1. `lon_min = -180; [-180]` STARTING LONGITUDE FOR X-AXIS
Sets the longitude of the left-hand edge of the plot.
2. `delta_lon = 90; [90]` INCREMENT OF LONGITUDE ON X-AXIS
Sets the longitude tick increment.
3. `contour_plot = 'n'; ['n']` OVERLAY CONTOUR PLOT?
Overlay line contours on the color block plot?
4. `contour_mod = 2; [2]` NUMBER OF COLOR INTERVALS PER CONTOUR
Number of color graduations per line contour.
5. `contour_mod_label = 4; [4]` NUMBER OF LABELED CONTOURS PER CONTOUR
Number of color graduations per labeled line contour.
6. `contour_label = 'y'; ['y']` LABEL CONTOURS?
Label the line contours (frequency of labeled contours set by `contour_label`).

¹⁸Note that if you did not run with ocean biogeochemistry (but rather climate-only), not all the plotting functions will run and you will have to restrict this default analysis to: `fun_make_analysis_phys('EXP1', 9999.5);`

¹⁹See 'help' on a specific plotting function for details of the relevant options in the parameter block.

7. `contour_noneg = 'n'; ['n'] RESTRICT DATA PLOTTED TO > 0.0?`
Restrict the plotted values to non-negative? (Can be useful if slightly negative values exist as can occur during tracer transport associated with large concentration gradients.)
8. `plot_log10 = 'n'; ['n'] PLOT LOG10 OF THE DATA`
Plot data values as log10(value)?
9. `contour_zero = 'y'; ['y'] PLOT ZERO CONTOUR`
Plot the zero contour?
10. `colorbar_old = 'n'; ['n'] PLOT 'OLD' COLORBAR`
Plot old style colorbar.
11. `data_offset = 0.0; [0.0] data offset (273.15 for K -> C)`
Introduce a data offset? This is useful for example for converting K to degrees C (removing the K value of 0 degrees C).
12. `data_ij = 'n'; ['n'] DATA as (i,j)?`
Overlay data in the form of (i,j) locations rather than longitude,latitude?
13. `data_ijk = 'n'; ['n'] DATA as (i,j,k)?`
Overlay data in the form of (i,j,k) locations rather than longitude, latitude, depth?
14. `data_ij_mean = 'n'; ['n'] average DATA by cell?`
Average overlay data per cGENIE grid cell rather than plotting raw locations.
15. `data_ijk_mean = 'n'; ['n'] average DATA by cell?`
Average overlay data per cGENIE grid cell rather than plotting raw locations.
16. `data_size = 25.0; [25.0] SIZE OF OVERLAY DATA POINTS`
Size of the overlay data points.
17. `data_anomaly = 'n'; ['n'] PLOT AS MODEL-DATA ANOMOLY ONLY?`
Plot data locations with the model-data anomaly rather than data value?
18. `data_only = 'n'; ['n'] PLOT ONLY DATA (no model values)?`
Plot only the overlay data locations (and not any model data)?
19. `data_site = 'n'; ['n'] PLOT DATA AS SITES (no data values)?`
Plot labeled site locations (no data value fill).
20. `plot_land = 'n'; ['n'] PLOT DATA OVER LAND?`
Plot data locations lying over land on the cGENIE grid (rather than screen out)?
21. `data_uv = 'n'; ['n'] overlay (u,v) velocity data?`
Overlay ocean current fields.
22. `data_uv_scale = 1.0; [1.0] scaling factor for vector length`
Scaling factor for velocity vectors.
23. `plot_opsi = ""; [""] PLOT OVERTURNING STREAMFUNCTION (basin)?`
Plot overturning streamfunction overlay?
24. `plot_opsi_min = -15; [-15]; plot_opsi_max = +15; [+15]; plot_opsi_dminor = 1; [1]; plot_opsi_dmajor = 5; [5]`
Controls on min, max and (major and minor) contour intervals.
25. `dscrsz = 0.60; [0.60] FRACTIONAL FIGURE WINDOW SIZE`
Adjustment factor of the fractional size (compared to the screen) of the figure window.

Further refinements: Examples

Examples:

1. To plot the positions (and labels) of data locations:

```
plot_fields_biogem_3d_k('cgenie_output','120926.SPIN','','49999.5,-1,'ocn_temp','','',
16,1.0,10.0,40.0,30,'','sites.dat')
```

where the experiment name is 120926.SPIN, the mapped variable is ocn_temp (although no model field need be plotted – set by an option in the plotting function itself, and the file of data locations is sites.dat.

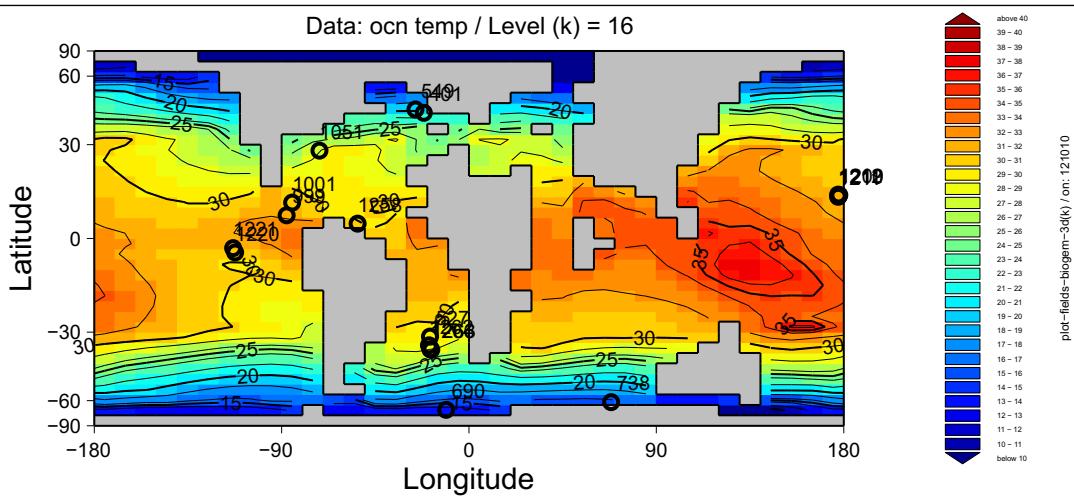


Figure 16.7: Paleocene-Eocene deep-sea sediment drill locations together with a contour-overlaid map of surface temperature.

16.3.4 Time-series plotting

16.3.5 Sediment model output analysis

17. muffindata

17.1 Data re-gridding

17.2 Miscellaneous data processing

muffingen basics

overview of the muffingen software

Installing muffingen

Running muffingen

Configuring muffingen – overview

muffingen configuration examples

Configuring muffin model experiments

muffingen parameter settings – details

18. muffingen

18.1 muffingen basics

18.1.1 overview of the muffingen software

muffingen is a collection of **MATLAB** functions that can create all the configuration files (excluding optional climatic and biogeochemical *forcing* fields) required by the cGENIE (**muffin**) Earth system model. **muffingen** is designed to take the output from a fully coupled GCM, particularly of past climates with different continental configurations, and re-grid the output needed by the **muffin** model, in the form of files of boundary conditions, all saved in their respective correct format. However, **muffingen** can also be used to draw conceptual alternative Earths (in terms of continental configuration). Again, saving the required **muffin** files in their appropriate formats.

As a **MATLAB** code, (only) very basic familiarity with using **MATLAB** at the command line is required. And a copy/license of the **MATLAB** software ... A sufficient grasp of **MATLAB** can be gained by going through the following sections/subsections of the matlabananas textbook, which can be found on [GitHub](#) (look for the PDF file compiled from the latex source – BANANAS.pdf):

- Section 1.1 – The **MATLAB** interface and command line.
- Section 1.6 – Changing directories and file search paths.
- Section 2.2 – Using *functions* ... of which **muffingen** is one.

18.1.2 Installing muffingen

The **muffingen** code is hosted on GitHub:

<https://github.com/derpycode/muffingen>

There are 2 ways to get your mitts on the **muffingen** code:

1. By downloading an archive file, containing all the code etc. For this – click on the green Clone or Download button, and select Download ZIP.
You then unpack/unzip the files and directory structure where you want it.
This [archive download] is a perfectly workable way to proceed¹ ...
2. Or you can *clone* the repository to where you intend to run **muffingen**. Note that you will need a git client installed on your computer. There are GUI clients for git, or this can be done at the command line:

```
$ git clone https://github.com/derpycode/muffingen.git
```

By doing this, you have created your own code repository (and an identical copy of the one hosted on GitHub). As part of the `git clone` command, you also automatically *check out* (from your very own personal repository) a copy of the code.

Note that you download or clone **muffingen** to the computer that you have **MATLAB** installed on and will use to run **muffingen** (i.e. not necessarily to the computer where **muffin** itself is run).

¹Note that this way, you will be unable to easily update the code with whatever new developments or bug fixes occur in the future, nor can propagate back any code changes that you might have made and might want to become part of the official **muffingen** code (i.e. downloading the zip file becomes a one-off installation that loses its formal connection to the GitHub repository).

18.1.3 Running muffingen

To use the **cGENIE.muffin** model configuration generator **muffingen** – at the command line in **MATLAB**, simply type:

```
>> muffingen('FILENAME')
```

where FILENAME is the name of an ASCII format configuration file (having a .m filename extension) that specifies the required settings (see subsequent section).

This file (FILENAME.m), must either be present in the directory where you installed **muffingen** (the directory where e.g. where the file **muffingen.m** is located) e.g. C:\muffingen. In this case, your **MATLAB** working directory must also be set to C:\muffingen. Or, the **MATLAB** working directory and the file FILENAME.m can reside elsewhere (both directory locations must be the same however). In this case, you will then 'add' the location of **muffingen** to **MATLAB**'s search path by the **MATLAB** function **addpath**².

The **muffingen** model configuration generator then starts, and depending on the specific settings in the configuration file, may require no user input, or may require user input, either because this option was requested, or because a re-gridding issue arose that requires manual intervention to resolve. A series of plots are created (and saved) as the configuration generation progresses together with the **muffin** model configuration files themselves. All the various steps plus details of how the contents of the configuration files are generated are reported at the command line, and saved in an ASCII format *.log file for future reference.

Depending on the specific configuration file settings (see later), **muffingen** has 4 main modes of operation, which are summarized as follows (and described in more detail, along with specific examples, below):

1. Configuration derivation based on re-gridding climate output from a GCM.

The most common usage of **muffingen**, and enabling a new (typically paleo) configuration to be derived from the output of a GCM experiment. Currently options for utilizing 4 different GCMs are provided: HadCM3(l), FOAM, CESM, and ROCKE-3D.

2. Derivation based on an existing model topography ('.k1') file.

Allowing a topography to be re-created, or adapted/altered, all directly from an existing model 'k1' file (and hence existing model configuration).

3. Derivation based on a prescribed land-sea mask.

This option will create a new configuration from any specified land-sea mask, whether 'real' or completely hypothetical.

4. From a blank (all ocean) initial template.

Finally, this option enables a topography to be 'drawn' within **muffingen** and hence represents an interactive alternative to (3) (rather than editing a land-sea mask in a text editor).

Also depending on the specific options, and particularly whether or not opt_user is set to true, and editor window will open, firstly for changes to the land-sea mask to be made, then later for the ocean bathymetry to be edited. Finally, a window will open allowing 'island paths' to be edited. The latter may be needed if **muffingen** cannot unambiguously determine them unaided.

²Refer to the **MATLAB** 'bananas' text for the syntax for using **addpath**.

18.2 Configuring muffingen – overview

When **muffingen** is run, a configuration file with a specified filename³ is loaded. The configuration file is in a simple plain text (ASCII) format, but is given a .m extension, enabling the values of a number of controlling parameter values to be set directly in **MATLAB**.⁴ The configuration file parameters control facets of **muffingen** behavior such as the primary model of operation, input and output filenames, what types of configuration files you want to generate, as well as there being a number of parameters controlling the finer details of re-gridding and configuration file generation, including whether to enable user-input or not.

The configuration files can be edited with the **MATLAB** editor (indeed, as a .m file, they are inherently a **MATLAB** format file), or any plain text (ASCII) editor. They must have a .m filename extension. The configuration file is divided up into a series of sections of different parameter options. The main (most commonly used) parameters are summarized as follows (and are more fully described later):

```
% *** CONFIG NAME AND MAIN INPUT SETTINGS ****
par_wor_name
Defines the name of the model configuration. This must be a string, 8 characters long.
e.g. par_wor_name='my_world'
par_gcm
Defines the format of the input. The string is blank for an interactive user-defined world,
i.e. par_gcm=''
See Section 16.1.3.
par_expid
Defines the name of the GCM experiment if a GCM input is selected, or the '.kl' or
'.dat' files if those respective input formats are selected by the above option.
By default, these files live in the INPUT.EXAMPLES sub-directory.

% *** FILE PATHS ****
% *** GCM netCDF FILENAMES ****
% *** GRID RESOLUTION ****
- par_max_i
Defines the number of grid points in the longitude ('i') direction.
This is typically 36 or 18, e.g. par_max_i=36
- par_max_j
Defines the number of grid points in the latitude ('j') direction.
This is typically 36 or 18 and is typically the same number as for the i direction, e.g.
par_max_j=36
- par_max_k
Defines the number of layers in the ocean circulation model.
This is almost always either 16 or 8, e.g. par_max_k=16
- opt_equalarea [default: true]
This specifies whether or not an equal area grid is used/assumed. (A value of false
results in the latitude grid being defined in equal increments of latitude).
```

³i.e. the single parameter passed to **muffingen** when it is invoked at the command line

⁴Note that the parameter filename is passed as a string without the .m extension (which is implicitly assumed). An error message will be generated if the file does not exist or has the incorrect extension.

```
% *** REGRIDDING SETTINGS ****
par_max_D [default: 5000.0]
Sets the maximum ocean depth (in m).

% *** CONFIG NAME AND MAIN INPUT SETTINGS ****
opt_user [default: true]
Determines whether muffingen pauses and allows user-modification of land-sea mask
and ocean depth (and island paths).
```

18.2.1 muffingen configuration examples

A series of example configurations are provided in the main **muffingen** directory. These illustrate parameter settings and the primary ways of using **muffingen**. They also serve as useful template parameter setting files. They are:

- `muffingen_settings_BLANK` – Starts from a blank ('water-world') all-ocean land-sea mask template. User input (allowing continents and seafloor topography to be 'drawn') is activated by default.
- `muffingen_settings_wppcont1` – Starts from a land-sea mask: `wppcont1.dat`, which is stored in the **muffingen** subdirectory `INPUT.EXAMPLES`. The mask defines an idealized pole-to-pole super-continent. There is no user input by default (so no modifications can be made, by default), but this can be enabled (`opt_user=true`).
- `muffingen_settings_drakeworld` – The first of a series of 4 conceptual worlds (loosely following the literature). As per `muffingen_settings_wppcont1`, this starts from a land-sea mask (`wordrake.dat`) which is stored in the **muffingen** subdirectory `INPUT.EXAMPLES`. The land fraction is minimal (as it is trying to reproduce the sort of zero-area numerical barrier used in the literature). This configuration has a barrier to ocean circulation, from the N pole down, with a high southern latitude gateway (a Drake Passage like feature).
- `muffingen_settings_eqpasworld` – As above, except with an equatorial (only) gateway.
- `muffingen_settings_ridgeworld` – As above, except with no gateway.
- `muffingen_settings_waterworld` – As above, but with no barriers, i.e. a 'water world'.
- `muffingen_settings_wor0251b1` – Start from a muffin model configuration 'k1' topography defining file: `p0251b.k1`, which is stored in the **muffingen** subdirectory `INPUT.EXAMPLES`. The k1 file defines a late Permian continental configuration (and bathymetry). User modification is enabled by default (so **MATLAB** pauses for modifications to be made), but this can be disabled (`opt_user=false`).
- `muffingen_settings_modern` – Takes the continental configuration and climate simulation output from a fully coupled GCM experiment (files in directory `xbowl` in the **muffingen** subdirectory `INPUT.EXAMPLES`) and derives full **muffin** model boundary conditions. User input (allowing continents and seafloor topography to be 'edited') is activated by default.

18.3 Configuring muffin model experiments

Having created a new **muffin** configuration using the **MATLAB** function **muffingen**, carry out the following steps:

1. Firstly, copy/transfer the entire configuration subdirectory (the directory with the same name as whatever you called your 'world') and its contents, from the **muffingen** output directory (e.g. **muffingen/OUTPUT.EXAMPLES**) on your local computer, to:

`cgenie.muffin/genie-paleo`
on the computer that you run the **muffin** model on.

2. Create a new *base-config* file. You do this by:

- (a) Taking one of the template *base-config* files:

`CONFIG_template_08lvl_R07.config`
`CONFIG_template_16lvl_C09.config`

The first being suitable for an 8-level (non seasonally forced ocean) and the second for a 16-level (seasonally forced) ocean.

Copy/rename the file to something ... 'appropriate' ... remembering: no spaces are allowed in the filename, and you need to retain the .config file extension.

- (b) There is a highlighted (<<< >>>) line in the template *base-config* file:

```
# ****
# GRID & BOUNDARY CONDITION CONFIGURATION
# ****
# insert the automatically generated muffingen parameter list here
# ****
# <<<                                >>>
# ****
```

Copy and past the contents of the **muffingen** output file⁵:

`config_yymmdd.txt`

into the template file where indicated (immediately above, immediately below, or simply replacing the (<<< >>>) line).

- (c) Now copy this new *base-config*, to:

`cgenie.muffin/genie-main/configs`

Note that in using one of the template *base-config* files, you have configured an ocean with just 2 tracers – temperature and salinity (i.e., there is no carbon cycle or ocean nutrients *etc.* enabled at this point).

3. Create a new *user-config* file to complete the experimental setup.

An example/template *user-config* file (designed for climate-only simulations) is also provided in the main **muffingen** directory:

`EXAMPLE.SPIN`

Rename and copy this (or copy and rename) to **muffin** model directory:

`genie.muffin/user-configs`

⁵In the filename, `yymmdd` is the date of the configuration creation.

18.4 muffingen parameter settings – details

UNDER CONSTRUCTION ...

Zonal wind-stress

For non GCM-based configurations, no prior wind fields exist. **muffingen** hence creates and configures an idealized zonal wind-stress field (from which wind velocity and wind speed is derived). The zonal wind-stress can take alternative strengths, depending on whether a high latitude gateway (in either hemisphere) exists. This can be prescribed directly, or **muffingen** can be enabled to 'choose' whether or not a high latitude gateway exists and hence whether or not to apply a strong or weak zonal flow. The parameter options (in the **muffingen** configuration file) are:

- `par_tauopt=0;`
muffingen chooses whether or not to apply a strong or weak zonal flow, and in which hemisphere.
- `par_tauopt=1;`
A weak zonal flow is applied in both hemispheres.
- `par_tauopt=2;`
A strong zonal flow is applied in both hemispheres.

For reference – the modern world has a mix of strong (southern) and weak (northern) hemisphere zonal flows.

The different EXAMPLE (.m) configuration files have a mixture of default choices:

- `muffingen_settings_BLANK,`
`muffingen_settings_drakeworld, muffingen_settings_eqpasworld,`
`muffingen_settings_ridgeworld, muffingen_settings_waterworld:`
`par_tauopt=0;`
- `muffingen_settings_wppcont1:`
`par_tauopt=2;`
- `muffingen_settings_modern:`
n/a (wind-stress derived from GCM fields)

(No EXAMPLE configurations currently specify a strong zonal field, although `muffingen_settings_ridgeworld` will be prescribed one automatically in **muffingen**.)

Installation related questions
Cluster/queue questions
Help! My experiment has died ... why?
Other sources of error
Meaning of specific error messages
General running and configuring experiments questions
Climate-y questions
Ocean biogeochemistry-y questions
Questions of long-term carbon cycling and stuff
Data saving questions
Forcings questions



19. FAQ

aka: 'Has this dumb question been asked before?'

19.1 Installation related questions

Stack space

You may encounter issues with regards to the ifort Intel FORTRAN compiler (an maybe others), particularly when using SEDGEM because of the size of the arrays holding sediment information:

"The Intel® Fortran Compilers 8.0 or higher allocate more temporaries on the stack than previous Intel Fortran compilers. Temporaries include automatic arrays and array sub-sections corresponding to actual arguments. If the program is not afforded adequate stack space at runtime relative to the total size of the temporaries, the program will terminate with a segmentation fault."

The (a?) solution is to increase the CPU stack space, Try:

```
$ ulimit -s unlimited
```

19.2**Cluster/queue questions**

Do I have to submit experiments to the queue rather than running interactively?

Yes! Except for developing the model and debugging, testing new experimental designs, and forcing a re-compile. The number of instances of the model that can be run simultaneously interactively is limited by the number of processing cores on the head node. The more experiments that are run interactively, the slower everything will go. Additionally, if you even temporarily lose your Internet connection, an interactively-run experiment will die. The queue is there for your convenience, believe it or not ...

Can I leave all my experiment results on the cluster for ever?

No! Nothing is backed up on the cluster, and space is not infinite. So, periodically, transfer archived (.tar.gz) results off of the cluster and delete both the archive file and the results directory.

19.3 Help! My experiment has died ... why?

If, when using the runmuffin.sh shell script to run a **cgenie.muffin** experiment, it all goes horribly pear-shaped ...

1. The experiment dies absolutely immediately.

Check that the runmuffin.sh shell script has executable permissions. Also check that the directory you are trying to run the model from is the genie-main directory.

2. The experiment does not quite die immediately, but does not manage to stagger even as far as the line:

```
>> Here we go ...
```

before dropping dead. If so, there should be an error message telling you that a particular file or directory cannot be found. Check:

- All the files and directories you have specified exist.
- You have not omitted spaces where you should not have, nor added spaces where a '_' separator was required.
- You have not misspelt anything – a common cause of problems is in reading the number one ('1') for the letter el ('l'), or *vice versa* in the computer font (*Courier*).

These first two sorts of pain and suffering are due to mis-configuration of the runmuffin.sh shell script. Also refer back to the overall sequence of configuring and running the **cGENIE.muffin** model shown in Figure 1.2.

19.3.1 Other sources of error

Other sources of error are due to the configuration of **cGENIE.muffin** (or more rarely, due to the model itself):

1. As **cGENIEmuffin** initializes, files may be reported as not being found. One possible cause of this is that '~~' may not necessarily get expanded into the path of your home directory (e.g., '/home/mushroom'. In this situation, '~~' can simply be replaced with '\$HOME'. Note that as well as making this substitution at the command line, the *user-config* file may also contain instances of '~~' (such as in specifying particular *forcings*).
2. A missing/not found error can also arise with some compilers if one of the various ASCII input files to **BIOGEM** (or **SEDGEM**) does not have a blank line at the bottom (some vague quirk of the unformatted read used in the **FORTRAN** code). Check: the *user-config* file, and also any boundary condition files being requested.
3. Further trouble can occasionally arise when using **WindoZ** and editing files (e.g., the *user-config* file) and it is possible to corrupt the format of the file. For what file(s) you have edited, use the command dos2unix to strip off **WindoZ** formatting characters (which are invariably invisible in most editors). The syntax for this (or see the **linux** 'man' pages, or even Google it) is \$ dos2unix FILENAME.
4. If the model starts running, but dies with a reported failure to solve the aqueous carbonate system, it may be that you need to force a re-compile (\$ make cleanall). Running **cGENIE.muffin** with array dimensions which do not match the number of tracers selected is a common cause of failures to solve the aqueous carbonate system, as often calcium ion or other tracer concentrations become corrupted and get assigned nutty and all but impossible values.

Also, if a *re-start* is used which was generated with a different land-sea mask (and *base-config*) to the current experiment and associated *base-config*. Try running without the *re-start* and see if that solves it (identifies the source of the problem).

19.3.2 Meaning of specific error messages

'ERROR: path integral around island too long'

Such an error is possible when developing new or modifying existing continental configurations (and associated 'island' and 'path' definition files), but not in normal running of the model. First try a `make cleanall` and then try re-running. If the problem persists, it is possible that a key configuration file has accidentally/somehow been changed. To check for this – do a `make cleanall`, and then from the `cgenie.muffin` directory:

```
svn status -u
```

Any file that you have modified is labeled with an `m`. Any new files on the server that you don't have will have a `*`. Files with a `?` are files that exist locally and are not on SVN (and can be ignored). If there is a file with an `m` that should not have been modified:

```
svn revert FILENAME
```

will re-set the file `FILENAME` (also include the relative path) it to the current SVN version status.

'ERROR MESSAGE: Particulate tracer CaCO₃ ...'

I have been told 'ERROR MESSAGE: Particulate tracer CaCO₃ does does not have the corresponding ocean tracer Ca selected' – is this a problem ... ?

No! You are simply being reminded that you have calcium carbon (CaCO₃) selected as a particulate tracer in the model, but although when it dissolves it releases Ca²⁺ (and removes Ca²⁺ when CaCO₃ is precipitated), you do not have Ca²⁺ selected as an explicit dissolved tracer in the ocean. This is not a problem as by far the most important effect on the carbon cycle of adding/subtracting Ca²⁺ is a change in alkalinity, which is implicitly account for. Only on **very** long time-scales, or in deep-time situations when the Ca²⁺/Mg²⁺ ratio was very different form today, might you need to select Ca²⁺ (and Mg²⁺) as an ocean tracer.

19.4**General running and configuring experiments questions****When does the model need to be recompiled?**

cGENIE.muffin will need to recompile in the following situations:

- You have just carried out one of the **cGENIE.muffin** tests, e.g., `make test` or `make testbiogem`.
- You have changed the dimension of the climate model grid (which also means an automatic change in the biogeochemistry modules), either horizontally (e.g., going from 36×36 to 18×18) or vertically (e.g., going from 8 levels in the ocean to 16).
- You have changed the number of selected ocean biogeochemical tracers in the *base-config* and hence changed the value of:

```
GOLDSTEINTRACSOPTS='$(DEFINE) GOLDSTEINTRACS=2'
```

(The latter two involve a change in compiled array dimension.)

In all three situations, the *base-config* is being changed (or should be¹). In running at the command line (i.e. interactively), the `runmuffin.sh` script detects the change in *base-config*, and automatically forces a re-compile for you. However, the compute nodes of the cluster do not have access to the **FORTRAN** compiler. As a sad and unfortunate consequence, submitted jobs cannot recompile modules and all science modules must be already compiled when a job is submitted.²

To recompile (and re link) the science modules – first, start an interactive run of the experiment you want to conduct. This will ensure that it is correctly compiled. This also serves as a visual check that you have requested a *user-config*, restart, etc that actually exists. Start the run for the length of time you intend to use when submitting the experiment as a job to the queue, but kill it (keyboard command: Ctrl-C) once it is compiled and you are happy that it is running OK (say, after 10 years). You can now be reasonably confident that the experiment is safe to submit the job to the cluster (and all files and inputs are as they should be).

If you have multiple experiments, all with the same resolution and number of tracers, you DO NOT need to re-run interactively or attempt to recompile. Also, you can add 'modules' and not recompile. i.e., you can interactively run an ocean -only carbon cycle. And then submit it. And then submit an experiment using **SEDGEM** as well. (Because when the model is compiled, ALL sciences modules are compiled, meaning that all there is to do is just link them, which does not require the (ifort) **FORTRAN** compiler.)

Refer to Figure 1.2 for the sequence of steps associated with configuring and running model experiments.

In the naming of different *forcing* specifications: what does 'yyyyz' mean?

A. The naming convention for *forcings* is that the (sub)directory name starts with the code for the continental configuration, if the *forcing* is tied to a specific continental configuration. For example: *forcings* with the string 'FeMahowald2006' relate to the prescription of a dust (Fe flux) field re-gridded from *Mahowald et al.* [2006]. When this has been re-gridded to the *worjh2* continental configuration, *worjh2* appears at the start of the name. If the *forcing* is independent of a specific continental configuration, such as restoring atmospheric CO_2 to a prescribed value (uniformly throughout the atmosphere), the string is 'yyyyz', as in e.g.: *pyyyyz_RpCO₂_Rp13CO₂*.

¹One could edit a *base-config* and re-ruin, but it is better to create a new *base-config* file if editing any of the settings, particularly those affecting array dimensions

²It is OK to change the flavor of GENIE as linking is done by the C compiler.

Can I make the model run faster?

sign You speed freak. Is this all you care about? What about the quality of the simulation - does that mean absolutely nothing to you? Oh well ... There is a bunch of stuff that slows **cGENIE.muffin** down that may not be absolutely essential to a particular model experiment. These include:

- The number of tracers - if you don't need 'em, then don't select 'em! Selected tracers are automatically passed to **GOLDSTEIN** and advected/conveyed/diffused with ocean circulation. Similarly, **BIOGEM** does a whole bunch of stuff with tracers, particularly those which can be biologically transformed. All this is numerically wasteful if you aren't interested in them. Equally importantly, the more tracers you have selected the more careful you have to be in configuring the model. Superfluous tracers therefore cost more configuration time and/or increase the chance of a model crash.
- *Tracer auditing* - the continuous updating and checking global tracer inventories to ensure that there is no spurious loss or gain of any tracer (i.e., a bug) has computational overheads associated with it. Whether this checking is carried out or not is set by the value of the flag `bg_ctrl_audit`³.
- *Time-series* results saving. Model tracer (plus some physical) properties are brought continuously averaged in constructing *time-series* results files. Cutting down on *time-series* that you don't need will help minimize model run-time. The various categories of time-series that will be saved are specified by a series of namelist parameter flags. However, within each category (such as ocn tracers - `bg_ctrl_data_save_sig_ocn`) all properties will be saved - you are not given the option to save a defined sub-set (for example, *DIC* and *PO₄* in the ocean but not *ALK*).
- Time-slice results saving. If you have relatively few requested time-slices over the course of the model integration then this is unlikely to significantly impact the overall run-time (even with all possible data category save namelist flags set to `.true.`). However, note that if you have accidentally triggered the default time-slice saving interval (by having no data items in the time-slice specification file (`bg_par_infile_slice_name`)) you may end up with the model running about as fast as a 2-legged dog super-glued to a 10-tonne biscuit.
- Alter the degree of synchronicity between climate and biogeochemistry (see HOW-TO guide).

As a very rough guide, the impact on total run-time of making various changes to the model configuration are listed as follows. Numbers are given as a percentage increase in total model run-time (using the `/usr/bin/time` linux command). Tracers selected in the ocean are DIC, ALK, PO₄, O₂, DOM_C, DOM_P, DOM_O₂, as well as 13C isotopic components (DIC_13C and DOM_C_13C) (+ T and S). The corresponding tracers are present in the atmosphere and as particulates. The model is run for 10 years as a new run (i.e., not loading in a restart file):

- ADD auditing $\Rightarrow +15\%$
- ADD time-slice saving $\Rightarrow +20\%$ ⁴

³It is `.false.` by default.

⁴Because only a 10 year integration has been carried out with a time-slice saved at 10 years, the computational cost of

- ADD time-series saving $\Rightarrow +15\%$
- REMOVE ^{13}C isotopic species (= DIC and DOC ocean tracers) $\Rightarrow -10\%^5$

You can also run at lower resolution. The basic configuration for a faster 'lego box' **cGENIE.muffin** configuration consists of a 18×18 model grid and an 8 level ocean. The continents are in a zonally-averaged configuration and there is no topography in the oceans.

The model is accelerated by:

1. it's low resolution
2. taking 48 instead of 96 ocean time-steps per year in the ocean
3. **BIOGEM** being only being updated every 4 rather than every 2 ocean time-steps.

In this configuration 100 years take about 40 seconds, 10 kyr would just take over and hour, and 100 kyr could be run overnight!

time-slice saving is disproportionately high as displayed. With a longer integration, the relative cost of saving a time-slice will fall. In contrast, the computational cost as a fraction of total run-time of time-series saving and auditing is likely to remain the same.

⁵The speed gained by removing two tracers is not proportional to the fractional decrease in number of tracers (in this example reducing from 11 to 9 the number of tracers in the ocean gives only a ca. 10% improvement in overall speed).

19.5 Climate-y questions

Can I disable climate feedback with CO_2 ?

Yes, for instance when you might want to compare the fate of CO_2 released to the atmosphere with climate (and ocean circulation and temperatures) not responding, vs. CO_2 in the atmosphere also driving changes in climate (and hence affecting the pathways and transformations of CO_2 , particularly in the ocean).

To specify no climate feedback, add to the *user-config* file:

```
# set no CO2 climate feedback  
ea_36=n
```

as well as a radiative forcing scaling:

```
# scaling for atmospheric CO2 radiative forcing, relative to 278 ppm  
ea_radfor_scl_co2=1.0
```

(a value of 1.0 giving no change in climate relative to the default).

How do I change the orbital configuration of cGENIE.muffin?

[see Orbits HOW-TO]

Can I do solar geoengineering (SRM) experiments?

No! Because you might destroy the planet.

No wait ... in the model (world) ... yes! But because of the absence of a dynamical atmosphere, options here are limited. However, modification of the solar constant, *a-la* 'giant mirrors in space' is possible. Sea-ice (surface) albedo can also be adjusted.

19.6 Ocean biogeochemistry-y questions

Can solubility related changes be separated from stratification and circulation changes?

With **BIOGEM** coupled to the climate model core of **cGENIE.muffin**⁶, a change in atmospheric CO_2 will induce a change in SSTs, which in turn affect the carbon cycle and feedback on CO_2 via changes in solubility and via changes in circulation (stratification) and thus biological productivity. There are times when it is helpful to separate out solubility related changes from circulation related changes. This equally applies to dissolved O_2 and CO_2 . The problem is that you need a change in climate and surface temperatures in the climate model in order to induce a change in circulation.

There is a way of having an altered climate and circulation, which then affects the marine carbon cycle, yet specify the SSTs that are actually seen by **BIOGEM** (and thus used in solubility calculations).

First of all, control the radiative forcing of climate internally in the **EMBM** rather than externally by the atmospheric CO_2 concentration calculated by **ATCHEM**. Turn off explicit CO_2 forcing of climate by setting: `ea_36='n'`. The namelist parameter `ea_20` will then dictate the EMBM radiative forcing: a value of 1.0 (default) gives no change in radiative forcing ($CO_2 = 278$ ppm), a value of 2.0 corresponds to the effect of doubling CO_2 , $\times 4 CO_2$, etc. Altering the value of `ea_20` thus lets you control climate (and circulation) without having to adjust CO_2 and the carbon cycle.

Next, SSTs in **BIOGEM** can be specified independently of the climate model. You achieve this by setting up a restoring forcing of ocean temperatures at the surface. Note that by default, prescribing SSTs (or SSSs) in **BIOGEM** does not propagate through to the climate model which does its own independent climate thing based on the value of `ea_20`. This allows you to retain the surface temperatures and thus solubility associated with a $\times 1 CO_2$ World, but have a warmer more stratified ocean (appropriate for a much warmer World).

What actually happens is that **BIOGEM** receives both the altered circulation field and the altered SSTs due to $\times 4 CO_2$, but sets its own SSTs internally rather than use those calculated by the climate model. Setting up the SST restoring is in principle just like for PO4. The values for the SST field you can simply copy and paste out of a prior $\times 1 CO_2$ experiment.

The converse experiment, is to have circulation and biological productivity not change, but explore the effect of changes in SST-driven solubility. i.e., to separate the solubility pump from circulation change effects on glacial CO_2 .

What is the difference between the different Fe configurations and schemes?

There is a parameter – `bg_opt_geochem_Fe` – that specifies the Fe scheme (although note that you have to have the appropriate/required tracers selected):

1. `bg_opt_geochem_Fe = 'OLD'`

This is the very original scheme, based on 3 tracers – Fe (free dissolved Fe), FeL (ligand-bound Fe), and L (free ligand).

During scavenging, a new equilibrium partitioning between the 3 species is calculated and imposed on tracer concentrations via the remin array, but only at the surface.

Entrain some completely unnecessary accounting for Fe addition at the surface as part of the equilibrium calculation, including a call to `sub_calc_geochem_Fe` as part of the main (i,j) biogem subroutine array. All other schemes are free from this.

2. `bg_opt_geochem_Fe = 'ALT'`

⁶Namelist: `ea_36='y'`

This is also based on 3 tracers – Fe (free dissolved Fe), FeL (ligand-bound Fe), and L (free ligand).

During scavenging, a new equilibrium partitioning between the 3 species is calculated and imposed on tracer concentrations via the remin array. This is carried out throughout the water column.

3. `bg_opt_geochem_Fe = 'FeFe2LFeL'`

Based on the same 3 tracers – Fe (free dissolved Fe), FeL (ligand-bound Fe), and L (free ligand), but also, Fe2, which is ignored for the purposes of calculating scavenging and Fe-FeL-L partitioning. Here, the tracer Fe is explicitly taken to be Fe^{3+} .

Scavenging and tracer concentration re-partitioning as per (2).

4. `bg_opt_geochem_Fe = 'hybrid'`

This is based on the minimum number of tracers required to describe the system – TFe (total dissolved Fe) and TL (total dissolved ligand). The equilibrium partitioning between Fe, FeL, and L, is derived when required.

5. `bg_opt_geochem_Fe = 'lookup_4D'`

Also based on TFe (total dissolved Fe) and TL (total dissolved ligand), but with Fe calculated via a lookup table.

In schemes 1+2+3, scavenged Fe is removed from the Fe tracer pool. For 4+5, scavenged Fe is removed from the TFe pool.

What is 'tracer auditing' – should I have it switched on?

When developing a new model parameterization, it is of fundamental importance that careful track is kept of the total tracer inventory of the system in the face of internal mass transfer and any inputs (e.g., prescribed restoring or flux boundary conditions) or outputs (e.g., sedimentation). No spurious gain or loss of tracer mass must occur as a result of bugs introduced to the code. The tracer inventories of the ocean can be periodically calculated and compared to that predicted to have occurred on the basis of any net input (or output) occurring in the intervening time to help catch bugs. The simplest implementation would be an audit carried out at system start-up (before any transformation of tracer mass has taken place), and at the very end (after the last transformation of the tracer fields). However, integrating over an extended time period can lead to the excessive accumulation of numerical (truncation) errors. Instead, the audits are carried out periodically during the model run. The periodicity of tracer auditing follows the times specified for time-series data saving (i.e., at time listed in the file specified by `bg_par_infile_sig_name`).

The entire audit procedure is as follows:

1. First, an initial inventory is calculated, achieved by summing the product of the concentration of each (selected) tracer with the mass of each cell, across all wet cells.
2. During the model run, the net spatially- and time-integrated transfer of tracer mass arising from all transfers across the external reservoir boundaries is calculated.
3. At a periodic pre-defined time, the inventories are re-calculated. The difference between old and new inventories should be equal to the integrated net flux. If the relative difference between re-calculated inventory and estimated (on the basis of net flux) differs by more than a predefined threshold then an error message is raised (and the model halted if requested)
4. The integrated net flux variable is re-set to zero and steps (2-4) repeated.

In short – if you are not modifying the code then you can take it on trust(!) that the model distribution is free of (major) bugs and that spurious gain or loss of tracers does not occur. If you don't trust me ... then switch the auditing feature on.

Auditing is inactivated by default. To activate it:

```
bg_ctrl_audit = .true.
```

To adjust the threshold (relative) tolerance⁷:

```
bg_par_misc_audit_relerr = value
```

To halt the model⁸ if it fails the tracer drift tolerance:

```
bg_ctrl_audit_fatal = .true.
```

A secondary benefit of tracer auditing when running the model interactively, is that it reports back to you the maximum and minimum value of all the tracers (and locations of where this occurs), as follows:

```
>>> SAVING BIOGEM TIME-SERIES @ year 0.50 278.069 -6.501 16.522 3.843 18.543 ...
temp / min = 0.2713E+03 (18,36, 8) / max = 0.3030E+03 ( 4,18, 8)
sal / min = 0.3337E+02 (10,35, 8) / max = 0.3891E+02 (30,29, 8)
DIC / min = 0.1878E-02 (35,24, 8) / max = 0.2581E-02 (33,21, 1)
DIC_13C / min = -.4225E+00 ( 3,16, 3) / max = 0.2792E+01 (25,13, 8)
DIC_14C / min = -.1779E+03 (33,21, 1) / max = 0.2197E+02 (30,29, 8)
PO4 / min = 0.7071E-07 (29,28, 8) / max = 0.3806E-05 ( 3,16, 3)
O2 / min = -.4521E-04 (27,30, 5) / max = 0.3363E-03 (24,35, 8)
ALK / min = 0.2212E-02 (10,35, 8) / max = 0.2724E-02 (33,21, 1)
DOM_C / min = -.4159E-05 (21,34, 3) / max = 0.1517E-04 (32,25, 8)
DOM_C_13C / min = -.1000E+20 ( 1, 3, 2) / max = 0.5817E+01 (29,36, 8)
DOM_C_14C / min = -.1000E+20 ( 1, 3, 2) / max = 0.2236E+04 (29,36, 8)
DOM_P / min = -.3924E-07 (21,34, 3) / max = 0.1431E-06 (32,25, 8)
Ca / min = 0.9769E-02 (10,35, 8) / max = 0.1136E-01 (30,29, 8)
CFC11 / min = 0.0000E+00 ( 1, 3, 2) / max = 0.0000E+00 ( 1, 3, 2)
CFC12 / min = 0.0000E+00 ( 1, 3, 2) / max = 0.0000E+00 ( 1, 3, 2)
Mg / min = 0.5050E-01 (10,35, 8) / max = 0.5888E-01 (30,29, 8)
```

How do I do an ocean CO_2 injection experiment?

There is a hard way (but maximum flexibility), a less hard way, ... and an easy way. To cut the shit – what follows is the easy way!

First, you want to use the updated tracer forcing format:

```
bg_ctrl_force_oldformat=.false.
```

Put this line in the *user-config* file if it is not already there, perhaps under FORCINGS section.

You will need a *forcing* template for the CO_2 injection – pyyyzz_FCO₂_UNIFORM. This is provided on mygenie.seao2.org. Download and unpack (`tar xfz pyyyzz_FCO2_UNIFORM.tar.gz`) from the directory: `~/genie_forcings`. As it stands, this is configured to stuff 1 PgC yr-1 of CO_2 into the ocean over the course of one year. The location of the CO_2 injection is some random default place that probably does not exist, which is not very good. So, you need to specify your ocean location. For this, add the following lines to a *user config* file:

```
bg_par_force_point_i=22
bg_par_force_point_j=33
bg_par_force_point_k=5
```

which corresponds to a cell in the N. Atlantic (i,j, = 22,33) at an intermediate depth (k=5).

The i,j,k coordinates are counted from left-to-right with longitude: i, from bottom to top with latitude: j, and from top to bottom with depth for ocean level, k. The land-sea mask and maximum depth (lowest k integer) you are allowed can be got from the BIOGEM 2D netCDF, variable

⁷By default, this is set to 1.0E-08.

⁸By default the model will continue running, even if there is an apparent spurious drift in tracer inventories occurring.

grid_level. This is a map of the 'k' values. >90 means land, for the 8-level ocean the ocean depths will be between 1 and 8. 8 being the surface. So the map is of the depth of the ocean and thus lowest k value you are allowed to use.

By default, using the CO_2 injection forcing template you will get 1 PgC emitted to the ocean, in the location you specify. You can scale the amount of carbon up via the namelist parameter:

```
bg_par_ocn_force_scale_val_3=xxx
```

where xxx is the multiple of 1 PgC you want to inject. NOT your favorite movie viewer rating. e.g., 100 PgC:

```
bg_par_ocn_force_scale_val_3=100.0
```

Note that 100.0 PgC is quite a lot of carbon to be injecting into a single location (cell) in the ocean model! By default, the time-scale of injection is set as 1 year. To increase the time over which the CO_2 injection takes place use the namelist parameter `bg_par_ocn_force_scale_time_3`, which simply scales the time interval. i.e.,

```
bg_par_ocn_force_scale_time_3=10.0
```

causes the CO_2 injection to take place over 10 years. But since the flux is in units of PgC per year, you will get 1000.0 PgC carbon total (10 years x 100 PgC yr⁻¹). So a combination of both namelist scaling parameters (both flux scaling, and interval scaling) will be needed for the required total CO_2 injection.

Note that the integer at the end of the namelist parameter name corresponds to the index of the ocean tracer. 3 is DIC. 12 would allow you to inject alkalinity into the ocean (but the you would need to create additional forcing specification files).

The slightly harder way involves entering in the i,j,k location explicitly in the forcing configuration file `configure_forcings_ocn.dat`. Altering the magnitude and/or duration of the flux release requires editing `biogem_force_flux_ocn_DIC_sig.dat`.

The hardest way requires that two 3D fields explicitly specifying the spatial nature of the forcing flux are created and modified.

For these alternative options – see earlier section on tracer forcings (Section 4).

Can I do carbon dioxide removal (CDR) experiments?

Yes! See geoengineering HOW-TO.

19.7 Questions of long-term carbon cycling and stuff

In GEMlite, does the adaptive step size control work with fixed/prescribed pCO_2 ?

If pCO_2 is fixed/restored, the answer is 'no' (ish). Or rather: you'll often get little difference compared to simply fixing the ratio of accelerated to non-accelerated time-steps. However, you will still get the advantage of adapting time-stepping depending on other changes to weathering (/sedimentation) that may have been prescribed. i.e. even with pCO_2 restored during 'normal' time-stepping, pCO_2 will change during the accelerated mode if weathering is significantly out of balance with sedimentation. The greater this imbalance, the greater the change in pCO_2 , and the sooner that time-stepping will be handed back to the normal (full updating) mode.

If you have prescribed changing pCO_2 , e.g. a continual ramp upwards, GEMlite is not appropriate in the first place, as the atmosphere is intrinsically assumed to be in equilibrium with the ocean surface and steady-state geochemical gradients in the ocean have been established. (This assumption is broken if CO_2 is rapidly invading the ocean.) Acceleration (and GEMlite) is also not appropriate if ocean circulation and carbon cycling have not yet been spun-up, unless at least 5 to 10 kyr of normal time-stepping forms part of the total spin-up including acceleration.

How can I diagnose changes in the carbon budget due to weathering/sedimentation?

The following example assumes that you are only running with $CaCO_3$ weathering (i.e silicate weathering and outgassing are both set to zero). In this case the weathering flux of DIC into the ocean is equal to the Ca weathering flux. This is output as a time series in:

`biogem_series_diag_weather_Ca.res`

in units of moles per year.

The system is closed with respect to organic matter, so that all POC is remineralised and returned to the ocean. For this reason, the exchange of DIC between the ocean and the sediments is equal to the exchange of Ca. i.e. the exchange of one mole of C is always associated with one mole of Ca, as the system is only open with respect to $CaCO_3$. Therefore the net flux of DIC from ocean to sediments is equal to the difference between `biogem_series_focnsed_CaCO3.res` and `biogem_series_fsedocn_Ca.res`.

The net flux of DIC into the ocean from weathering and sediments is therefore equal to `weather_Ca + fsedocn_Ca - focnsed_CaCO3`.

19.8 Data saving questions

Why is the netCDF data saved at odd times?

There is a default sequence of points in time that **BIOGEM** will save data at. These points are specified in the file: `save_timeslice.dat` (which lives in `cgenie.muffin/genie-biogem/data/input`).⁹ This default sequence provides a useful generic starting point.

To specify different save points for an experiment:

1. Edit this file (not recommended).
2. Copy, or create a new file (with the same format). The file that **BIOGEM** uses for saving data is specified by the parameter:

```
bg_par_infile_slice_name='save_timeslice_historicalfuture.dat'
```

(in the example of a historical/future relevant series of save points being requested).

Note that always, at the very end of an experiment, data is automatically saved regardless of whether or not you remembered to specify a save point for the final simulated year.

Refer to the Chapter on **cGENIE.muffin** model output.

⁹As a default, the netCDF *time-slices* are saved as annual averages, centered on these points in time.

19.9 Forcings questions

Can I combine *forcings* together?

Yes ... but it is not quite as simple as in the *user-config* writing:

```
# specify forcings
bg_par_forcing_name="worjh2.FeMahowald2006"
bg_par_forcing_name="pyyyyz.FRpC02_Fp13C02"
```

in the example that you wanted to combine an atmospheric CO_2 emissions *forcing* with a surface ocean dust *forcing*, because only the last parameter value in a list of multiple definitions is used. i.e. the above is equivalent to just writing:

```
# specify forcings
bg_par_forcing_name="pyyyyz.FpC02_Fp13C02"
```

Instead, you need to create a new forcing (assuming the combined forcing you want does not already exist):

1. Copy/rename one of the two individual *forcing* directories. This will be your new *forcing* name.
2. In the example above – if you have copied the directory for *worjh2.FeMahowald2006*, you simply need to add in the specific atmospheric CO_2 emissions *forcing* forcing files contained in *pyyyyz.FRpC02_Fp13C02*, which are:
`biogem_force_flux_atm_pCO2_13C_sig.dat`
`biogem_force_flux_atm_pCO2_sig.dat`
`configure_forcings_atm.dat`

Note that more care has to be taken when combining *forcings* that include the same phase of tracer, i.e. atmosphere and atmosphere, or ocean and ocean. In this case, you need to open up the *configure_forcings_*.dat* file of one *forcing*, and copy the tracer selection line (or lines)¹⁰ to the equivalent file in the new *forcing* directory.

Why does my ocean flux forcing does not do anything?

As always if you apply a flux forcing and nothing appears to happen, check:

1. The flux has not been scaled to zero ...
2. The spatial locations, where you expect the flux to be applied, and not on land ((i,j) location is a land, not ocean point), or in the ocean crust ((i,j) is ocean, but the layer chosen is deeper than the ocean floor at that location).
3. That the model run, in time, overlaps with the time-dependent forcing information. e.g. you might start a forcing at year 2010, but only run the model to year 2000 ...

Careful comparison, e.g. difference maps or simply looking at some global diagnostic output provided as in the time-series data format, will confirm whether the impact truly is zero, or just very small. If very small, your issue is mostly simply one of scaling and too small of a flux to make much impact.

¹⁰These occur between a pair of tags:

-START-OF-DATA-
-END-OF-DATA-

Why does my ocean iron flux forcing does not do anything?

Start by referring to above (general flux forcing question).

However, there is a special point of failure of a forcing, unique to the iron system, because there are two different ways of representing *Fe* and *Fe*-related species in **cGENIE.muffin**:

1. In the basic, and original Fe scheme, there are three separate tracers represented in the ocean:

Fe – tracer number 09 – dissolved iron III (Fe).

FeL – tracer number 23 – ligand-bound Fe.

L – tracer number 24 – free ligand (iron binding).

In the forcing definition, a flux of Fe is selected in:

`configure_forcings_ocn.dat`

by:

`-START-OF-DATA-`

`9 f f 0.0 t t -1 01 01 01 '[Fe]',`

`-END-OF-DATA-`

Associated with this selection, is a field of time-dependent information for the forcing:

`biogem_force_flux_ocn_Fe_sig.dat`

and dependent on the nature of the forcing, potentially also a file containing a spatial pattern for the forcing, e.g.

`biogem_force_flux_ocn_Fe_SUR.dat`

Here: it is important to note that both file names contain the tracer short-name: **Fe**.

2. In a newer scheme, there are just 2 tracers:

TDFe – tracer number 90 – total dissolved Fe.

TL – tracer number 42 – total dissolved ligand.

(and e.g. free iron is derived by assuming an equilibrium partitioning based on the total iron and total ligand concentrations).

Why am I telling you all this? For example, configurations using **ECOGEM**, use the newer (two tracer only) representation of iron cycling, whereas in e.g. geoengineering examples, **BIOGEM** is using the older three tracer representation. If you then wish to configure **ECOGEM** to use forcings based on the geoengineering examples, you have to:

1. Firstly, in `configure_forcings_ocn.dat` change the selected tracer number from 9 to 90.
2. Secondly, rename the time-dependent information file, and if present, the spatial file, changing the **Fe** bit of the filename to TDFe.
3. Lastly, the parameter in the user-config that scales the forcing (if used), has a name that ends in the tracer number and needs to be changed, so rather than e.g.

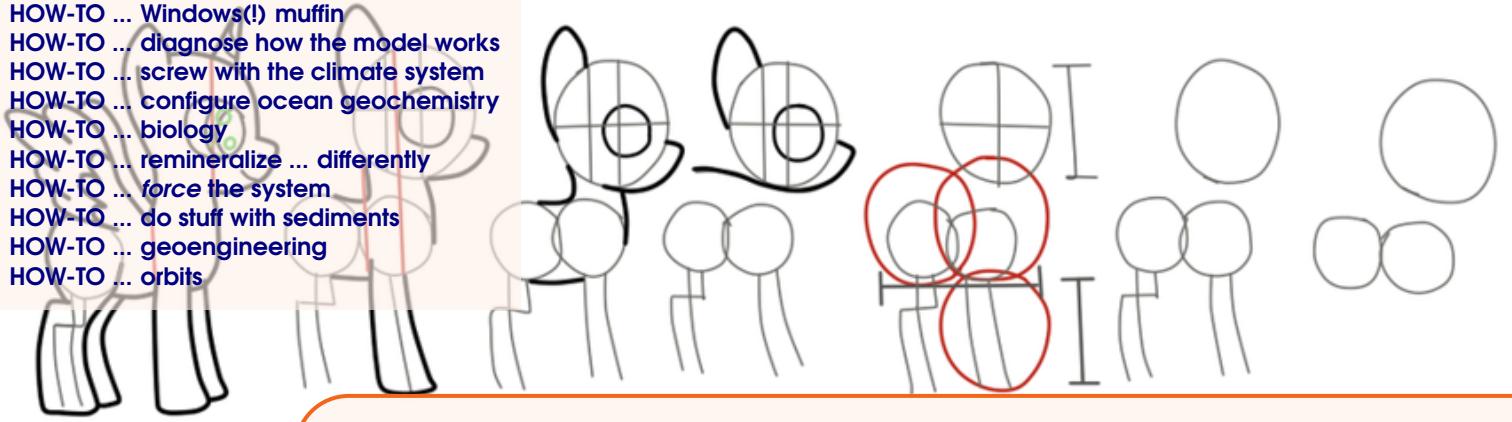
`bg_par_ocn_force_scale_val_9`

you would have:

`bg_par_ocn_force_scale_val_90`

If you select a tracer number in the *forcings* that does not exist in the ocean configuration you are using, such as the 'wrong' iron tracer – this is why the *forcing* appears not to do anything.

HOW-TO ... get started with cGENIE.muffin
HOW-TO ... linux
HOW-TO ... git
HOW-TO ... Sun Grid Engine
HOW-TO ... Ubuntu muffin
HOW-TO ... RedHat muffin
HOW-TO ... macOS muffin
HOW-TO ... Windows(!) muffin
HOW-TO ... diagnose how the model works
HOW-TO ... screw with the climate system
HOW-TO ... configure ocean geochemistry
HOW-TO ... biology
HOW-TO ... remineralize ... differently
HOW-TO ... force the system
HOW-TO ... do stuff with sediments
HOW-TO ... geoengineering
HOW-TO ... orbits



20. HOW-TO

What follows are potted HOW-TO instructions for doing things.
There is some overlap with the FAQ Chapter, so please read both!

20.1 HOW-TO ... get started with cGENIE.muffin

Install cGENIE

See: cGENIE *Quick-start Guide*. (Also refer to the READ-ME file for e.g., details of changes in configuring and running cGENIE compared to GENIE.)

Find configurations for cGENIE

A series of (example) cGENIE configurations are provided, many of which are detailed in full in the cGENIE *Tutorial* document. Example configurations comprise *base-config* and *user-config* files, plus any *forcings* needed.

- cGENIE *base-configs* are stored in:
/cgenie/genie-main/configs
and all start with 'cgenie_', for example:
cgenie_eb_go_gs_ac_bg_hadcm31_eocene_36x36x16_2i_080928_BASE.config
- cGENIE user configs are stored in:
/cgenie/genie-userconfigs
- cGENIE forcings are stored in:
/cgenie/genie-forcings

Do some thing dumb

Easy! Just close your eyes and change some parameter values at random. Better still, start using the model without reading the manual first ...

20.2 HOW-TO ... linux

Viewing directories; moving around the file system

When logging in, you always start from your 'home' directory. This is represented by a '~~' before the command prompt (\$). At the command line prompt in linux, you can view the current directory contents:

```
$ ls
```

or for a more complete output:

```
$ ls -la
```

To go down a directory (e.g. cgenie_output) relative to where you already are:

```
$ cd cgenie_output
```

and to go back up one is:

```
$ cd ..
```

It is often safer/easier at first, if you need to change more than one directory level to do this in stages. e.g. to change to cgenie_output/exp0_modern_SPINUP, change to cgenie_output (cd cgenie_output) but then check that you are in the place you think you are and/or remind yourself of the spelling of the next directory you need to change to by typing ls.

You can always return to your home directory (~) by typing:

```
$ cd  
$ cd $HOME
```

(or cd ~)

Copying and moving files

To copy a file myconfig to myconfig_new, assuming you are in the same directory where both the old file is and the new file will be:

```
$ cp myconfig myconfig_new
```

To move myconfig to the cGENIE user-config directory, assuming you are in the directory where the old file is but with the new file in a different directory, give the full path of the new directory:

```
$ mv myconfig ~/cgenie.muffin/genie-userconfigs/LABS/myconfig
```

To rename myconfig to useless_config:

```
$ mv myconfig useless_config
```

Creating directories

To create a directory mydirectory:

```
$ mkdir mydirectory
```

Repeating command lines

You do not have to re-enter lines of commands and options in their entirety each time – by pressing the UP cursor key you get the last command you issued. If you keep pressing the UP cursor key you can recover progressively older commands you have previously entered. When you have recovered a helpful line you can simply just edit it, navigating along with the LEFT and RIGHT cursor keys (press RETURN when you are done).

vi

The **vi** editor is a text-based editor that you use at the command line (i.e. it does not open in its own window, nor have fancy menu items or icons to click). You use **vi** = to create/edit files, by typing:

```
$ vi FILENAME
```

where **FILENAME** is the ... name of the file you want to edit¹ or create.

You start in the 'command' mode, in which you do not edit the contents directly, but instead can access file and copy-paste operations such as:

- :q == quit
- :q! == no, really quit
- :x == save and quit
- dd == cut line
- p == paste

To start editing and e.g. inserting text, press the **i** key.

To exit the editing mode and back to the command mode, press the **Esc** (escape) key.

¹Then make sure the file is present in the directory you are currently in, or provide a full path to the file (+filename).

20.3 HOW-TO ... git

20.4 HOW-TO ... Sun Grid Engine

For submitting jobs using **Sun Grid Engine (SGE)** on a cluster, a basic command² would look like this:

```
$ qsub -S /bin/bash runmuffin.sh <options>
```

Here: the `-S /bin/bash` part is to ensure that **SGE** uses the BASH shell to submit the job because this is the language that `runmuffin.sh` has been written in.

Take care that the installed **FORTRAN** compiler can be seen by the cluster nodes. If not, the **muffin** executable must have already been built prior to submitting a job. The easiest way to do this is to run **muffin** interactively briefly (e.g., with a run length of just a couple of years or kill it) and then submit the full run to the cluster.

Other useful submission options for **SGE**:

- To redirect the standard output stream:

```
$ qsub -o genie_log ...
```

- To redirect the standard error stream:

```
$ qsub -e genie_log ...
```

- To merge standard output and error streams into standard output:

```
$ qsub -j y ...
```

- To specify particular resources, such as the nodes with 8 GB of RAM:

```
$ qsub -l mem_total = 8.0G ...
```

- To decrease the priority of a job³:

```
$ qsub -p 1 ...
```

- To submit a job from the current working directory:

```
$ qsub -cwd
```

- Request an email is sent when the job starts and/or when it finishes – see the main pages for `qsub` for the required syntax).

- A complete example for the domino UCR clusters would be:

```
$ qsub -q dog.q -j y -o cgenie_log -V -S /bin/bash runmuffin.sh
      cgenie.eb_go_gs_ac_bg.worjh2.ANTH / EXAMPLE.worjh2.Caoetal2009.SPIN 10000
```

which merges standard output and error streams and redirects the resulting file to the directory `~cgenie_log`. Note that to redirect output as per in this particular example, the directory `~cgenie_log` **MUST** be present. I have no idea what happens if it is not ... but it can't be good ;)

You can check the status of the **SGE** job queue⁴ with the command:

```
$ qstat -f
```

and you can kill a job with the `qdel` command, the job numbers being given by the `qstat` command.

²With a different queue management environment it may be necessary to place the call to `runmuffin.sh` together with its list of parameters into an executable shell, and submit that instead.

³The default priority is 0. A lower priority has a higher value ... !

⁴Depending on the cluster setup, it may be possible to graphically check what is going on via the www.

20.5 HOW-TO ... Ubuntu muffin

This is a brief guide to installing **muffin** on **Ubuntu**. The example taken is for a generic **Ubuntu** distribution version 18.04 LTS ('Bionic Beaver') install that is fresh out of the box. For a more established installation, fewer components may need to be installed. Different/fewer components may also be necessary for an older **Ubuntu** distribution. The **netCDF** installation procedure has also been tested on **Ubuntu** 16.04 LTS and checks out A-OK.

Instructions are given step-by-step, although not all the components need be installed in this order. In brief: required installation components include: a **fortran** compiler, a **git** client, **netCDF** libraries. Note that the various **netCDF** component version numbers are not the most up-to-date, and newer can almost certainly be substituted (but not tested here).

Preparation

Get hold of a computer with **Ubuntu** installed on it. Make sure you have plugged in the internet connection. Log in. Get a strong cup of coffee.

Installation

1. **gfortran** [FORTRAN compiler]

```
sudo apt install gfortran
```

2. **g++** [C++ compiler]

And ... also the **GNU C++ compiler**⁵:

```
sudo apt install g++
```

3. **netCDF** [graphics libraries]

These come rather inconveniently in 2 parts ... first the **C** libraries need to be installed, and then the **FORTRAN** libraries

The example is given for the most recent version of both libraries. For details/most recent version, see: <https://www.unidata.ucar.edu/software/netcdf/>

But first ... missing in the default **Ubuntu** 18.04 is the **m4** utility, which can be installed:

```
sudo apt install m4
```

and also **make**:

```
sudo apt install make
```

Now you are good to go ... First, we need install the main **netCDF C** libraries and then the **FORTRAN** and **C++** libraries that depend on the **C** libraries. For the basic **C** libraries:

```
wget ftp://ftp.unidata.ucar.edu/pub/netcdf/netcdf-4.6.1.tar.gz
tar xzf netcdf-4.6.1.tar.gz
cd netcdf-4.6.1
./configure --disable-netcdf-4 --disable-dap
make check
sudo make install
```

⁵This could probably be worked-around by editing some of the **make** files ...

Note that this installs the libraries in the default install location. Also disabling **netCDF-4** to avoid added complexities of needing an **HDF5** library built with **zlib** enabled (and hence the **zlib** library installed) ...

Then ... **muffin** requires **netcdf-cxx** (legacy) C++ libraries for **netCDF** that have since been retired⁶ and hence requiring a sperate/additional installation step⁷ (first return back one directory level to where-ever you are downloading to and installing from):

```
wget ftp://ftp.unidata.ucar.edu/pub/netcdf/netcdf-cxx-4.2.tar.gz
tar netcdf-cxx-4.2.tar.gz
cd netcdf-cxx-4.2
./configure
make check
sudo make install
```

And ... then for the **FORTRAN netCDF** libraries. (At this point – return back one directory level to where-ever you are downloading to and installing from.)

```
wget ftp://ftp.unidata.ucar.edu/pub/netcdf/netcdf-fortran-4.4.4.tar.gz
tar xzf netcdf-fortran-4.4.4.tar.gz
cd netcdf-fortran-4.4.4
./configure
make check
sudo make install
```

NOTE: if for either or both libraries, you want them anywhere other than off of `/usr/local`, you'll need to specify the installation directory, set `NCDIR`, and pass this to `./configure --prefix=${NCDIR}`. And probably engage in additional unpleasantness.

Finally – if when running `make testbiogem` you run into issues, specifically: libraries that cannot be 'found', try forcing an update of the library link cache (the not found libraries may be links to the 'real' library and somehow this link is not working/found):

```
sudo ldconfig
```

[’creates the necessary links and cache to the most recent shared libraries found in ... the file `/etc/ld.so.conf`, and in the trusted directories, `/lib` and `/usr/lib`]’

4. **xsltproc** [*a command line tool for applying XSLT stylesheets to XML documents*]

```
sudo apt install xsltproc
```

Why do we eve need this? :(

5. **git** [git client]

You’ll need a git client:

```
sudo apt install git
```

⁶See: [link](#)

⁷This version of the netCDF C++ library includes no changes since the 4.1.3 release, but is provided for backwards compatibility as a separate package. It was developed before key C++ concepts like templates, namespaces, and exceptions were widely supported. It’s not recommended for new projects, but it still works.’

6. muffin [the cGENIE.muffin code]

And finally, you can obtain the code.

NOTE: Recommended is changing directory to your HOME directory (`cd`) as the default **muffin** directory settings assume this.

```
git clone https://github.com/derpycode/cgenie.muffin
```

7. python [python symbolic link]

If an actual **muffin** experiment, rather than just `make testbiogem`, does not run, and e.g. in response to:

```
which python
```

you get nothing (although you still get a response to `python -V` or type `-a python`), it maybe that you need a python symbolic link. It might also then not hurt to install `python2.7`

...

```
sudo apt install python2.7
```

to install `python2.7`, and:

```
sudo ln -s /usr/bin/python2.7 /usr/bin/python
```

to create a symbolic link from `python` → `python2.7`

Obviously, you are going to have configured **muffin** and got it running first (see next section) to even find out whether you need to muck about with `python` or not ...

Configuring muffin

The version of **muffin** available from GitHub is configured with default environmental settings that match the above instructions and require no configuration changes. If not – there are several environment variables that may need changing – the compiler name, netCDF library name, and netCDF path. These are specified in the file `user.mak` (`genie-main` directory). If the **muffin** code tree (`cgenie.muffin`) and output directory (`cgenie_output`) are installed anywhere other than in your account HOME directory, paths specifying this will have to be edited in: `user.mak` and `user.sh` (`genie-main` directory). If using the `runmuffin.sh` experiment configuration/launching scripts, you'll also have to set the home directory and change every occurrence of `cgenie.muffin` to the model directory name you are using (if different). (Installing the model code under the default directory name (`cgenie.muffin`) in your `$HOME` directory is hence by far the simplest and avoids incurring additional/unnecessary pain (configuration complexity) ...)

Testing muffin

To test the code installation – change directory to `cgenie.muffin/genie-main` and type:

```
make testbiogem
```

This compiles a carbon cycle enabled configuration of **muffin** and runs a short test, comparing the results against those of a pre-run experiment (also downloaded alongside the model source code). It serves to check that you have the software environment correctly configured. If you are unsuccessful here ... double-check the software and directory environment settings in `user.mak` (or `user.sh`) and for a **netCDF** error, check the value of the `NETCDF_DIR` environment variable. (Refer to the FAQ section for addition fault-finding tips.) If environment variables are changed: before re-trying the test, you will need to type:

```
make cleanall
```

That is for the basic installation.

20.6 HOW-TO ... RedHat muffin

No specific example is given at this time for **RedHat** ... refer to the installation instructions for **Ubuntu** in Section 20.5. Probably, not more installation will be needed. The **netCDF** requirements are the same.

20.7 HOW-TO ... macOS muffin

This is a brief guide to installing **muffin** on a Mac.

To install the **muffin** release of cGENIE on a Mac you will need a number of software packages, including Fortran, C++ and NetCDF. The best way to get hold of these is via a package management system, such as Homebrew (<https://brew.sh>) or MacPorts (<https://www.macports.org>). This guide is based on Homebrew, because it is slightly more user friendly than MacPorts, and is kept more up-to-date with changes in the Apple operating system. (If you are already a MacPorts user, please see Section 20.7).

1. First of all, you will need XCode, which can be downloaded from the app store, or here...

<https://developer.apple.com/xcode/downloads>

After installing XCode, it is necessary to enable command line tools, by entering at the command line...

```
xcode-select -install
```

2. Get Homebrew by pasting the following at the terminal command line...

```
/usr/bin/ruby -e
```

```
"$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

(all one line)

Next, type ‘brew doctor’. This should tell you “your system is ready to brew”. If it doesn’t, see Section 20.7.

3. Install Fortran, C++, NetCDF and some other useful libraries at the command line (using Homebrew) as follows:

```
brew install cmake
brew install gcc
brew install hdf5
brew install netcdf
brew install wget
```

4. Get hold of a current copy of the **muffin** code:

```
git clone https://github.com/derpycode/cgenie.muffin
```

5. Check your netcdf version number by entering `brew info netcdf`

This will return several lines, but the key one gives the netcdf path, and should look something like:

```
/usr/local/Cellar/netcdf/4.6.1_2 (84 files, 6.2MB) *
```

(This was from my most recent install, with version 4.6.1_2)

6. Finally, adjust the cGENIE environment variables for your machine and netcdf installation by editing

`cgenie.muffin/genie-main/user.mak`, setting:

```
MACHINE=OSX
```

and

```
NETCDF_DIR=/usr/local/Cellar/netcdf/4.6.1_2
```

to reflect your netcdf path (Step 5).

7. To test the code installation – change directory to `cgenie.muffin/genie-main` and type:
`make testbiogem`

This compiles a carbon cycle enabled configuration of *cGENIE* and runs a short test, comparing the results against those of a pre-run experiment (also downloaded alongside the model source code). It serves to check that you have the software environment correctly configured. If you are unsuccessful here ... double-check the software and directory environment settings in `user.mak` (or `user.sh`) and for a netCDF error, check the value of the `NETCDF_DIR` environment variable. (Refer to the User Manual for addition fault-finding tips.) If environment variables are changed: before re-trying the test, you will need to type:

```
make cleanall
```

That is is for the basic installation.

MacPorts

You are here because you have already installed MacPorts, presumably by following the instructions here:

<https://www.macports.org/install.php>. You now have two options, either remove MacPorts entirely, replacing it with Homebrew, or install the required packages through MacPorts and a number of precompiled binaries.

Option 1: Remove MacPorts from your system... (to be replaced by Homebrew)

1. Back up your system (i.e. using Time Machine).
2. To uninstall MacPorts, enter at the terminal:

```
sudo port -f uninstall installed
```

Then remove everything that is left from MacPorts:

```
sudo rm -rf /opt/local  
sudo rm -rf /Applications/DarwinPorts  
sudo rm -rf /Applications/MacPorts  
sudo rm -rf /Library/LaunchDaemons/org.macports.*  
sudo rm -rf /Library/Receipts/DarwinPorts*.pkg  
sudo rm -rf /Library/Receipts/MacPorts*.pkg  
sudo rm -rf /Library/StartupItems/DarwinPortsStartup  
sudo rm -rf /Library/Tcl/darwinports1.0  
sudo rm -rf /Library/Tcl/macports1.0  
sudo rm -rf ~/.macports
```

Note that the `sudo` command is inserted before the `rm` (i.e. remove) command in order to enable the correct permissions.

3. You may now continue with your installation as described in the main text. You may have to delete some files (using `sudo rm`), as recommended by `brew doctor`.

Option 2: Install required packages through MacPorts and precompiled binaries...

1. First of all, synchronize your installation of MacPorts:
`sudo port -v selfupdate`
2. Then install Netcdf and related C++ and Fortran libraries at the command line using MacPorts, as follows:
`sudo port install netcdf`

```
sudo port install netcdf-cxx
sudo port install netcdf-fortran
```

3. Download precompiled fortran and C⁺⁺ binaries appropriate to your operating system (El Capitan & Sierra, etc.) from <http://hpc.sourceforge.net>. Install as follows, amending the file number to match your version of OSX (e.g. El Capitan & Sierra are associated with the 7.1 binaries).

```
cd ~/Downloads/
gunzip gcc-7.1-bin.tar.gz
sudo tar -xvf gcc-7.1-bin.tar -C /
gunzip gfortran-7.1-bin.tar.gz
sudo tar -xvf gfortran-7.1-bin.tar -C /.
```

If your operating system is not listed here, you will either have to wait until it is, or install Homebrew.

Errors

Errors identified by ‘brew doctor’ are most likely associated with some incompatible files in your software libraries, perhaps from a previous installation of MacPorts. Try to follow the suggestions given to you by ‘brew doctor’, deleting any problematic files (using `sudo rm` to overcome any permission issues). Note that you may wish to do a system backup first.

20.8 HOW-TO ... Windows(!) muffin

Natively ... **muffin** does not compile under Windows. However ...⁸

Cygwin

So you want to run **muffin**, but previous sage advice has failed to deter you from wanting to do it on Windows... Well don't say we didn't warn you, but it *is* possible to do – but only be cheating.

muffin is not able to run directly on Windows, but there are ways to *pretend* that Windows is actually Unix and get it going anyway. But why oh why run **muffin** in Windows? Big runs are best sent to clusters (which are invariably a Linux machine of some sort) where they can happily run in native Linux and keep at it for ages. But maybe you want to tinker around with the model on your own laptop a bit and do some short runs without bothering with clusters and queuing and the like. Or maybe you don't have access to a fancy cluster and want to play with a real earth system model on your laptop anyway. Or maybe the normal way of doing it was just too easy for you...

The main method described here uses **Cygwin**, which claims to “provide functionality similar to a Linux distribution on Windows”. Alternatively, if you have Windows 10 you could try using its new-ish Windows Subsystem for Linux (WSL) feature, which in theory means you can have supposedly actual Linux on your Windows machine without having to do anything awkward like dual-installation. However, it's still a bit Beta, and when I last tried it put the requisite libraries all over the place which was a faff. It may work better at some point, but for the moment proceed there at your own risk (or do a better job than me).

Cygwin goes to pains to note that it only provides functionality similar to Linux, and isn't actually Linux or “... a way to run native Linux apps on Windows”. What this means in practice is that you can't just take stuff compiled on a Linux machine, dump it on your Windows machine and expect it to work via **Cygwin** (or vice versa). But if you're coding and compiling stuff to run within **Cygwin** then it basically acts the same. Just remember the golden rule: **What Compiles In Cygwin, Stays in Cygwin**.

We'll assume you don't already have **Cygwin** installed (and if you do, you may need to check you've got all the packages you need anyway – skip to running the installation executable from wherever you saved it). It's best to use this guide in parallel to the main **muffin** manual too (which provides extra troubleshooting). Here goes:

1. **Download Cygwin:** Head to <https://www.cygwin.com/> and click the link named `setup-x86_64.exe` under `Install` to download the executable (pick 64/32 bit depending on your system). Save it – maybe adding **Cygwin** to the name so you know what it is later – and then run it from your downloads tab when it's ready (it will also no doubt have a security popup; say yes, or this will be a very short tutorial)
2. **Start Cygwin Setup:** In **Cygwin** setup, say `Next` to all the things [assuming you want to install from the internet (rather than somewhere else?), use the default root directory (best not change this), install for all users on your machine, and default local package directory, & use system proxy settings for downloading]. It will ask you what mirror you want to download from too – it doesn't really matter where, I tend to go for somewhere local but they should all be the same anyway

⁸By Dr. David A. McKay <david.armstrongmckay@su.se>, tested May 2019

3. **Select Packages:** Next in Setup, it will present you with Select Packages to install. **Cygwin** doesn't automatically come with every useful Unix package and library ever (as it'd be enormous), so you have to go through the tedium of picking what you actually think you need (and inevitably miss something). Here is what you'll need for **muffin** install to work beyond the default Basics (find them using the search bar, then change the dropdown bit where it says 'Skip' to the latest non-test version number. Libraries for each package should be automatically selected by this as well):

- (a) gcc-core; gcc-fortran; gcc-g++ – compilers for c, c++, fortran for compiling the model
- (b) nano – a standard text editor (or your favourite Unix one, e.g. emacs or vim) so you can edit
- (c) git – for downloading/cloning **muffin** (and updating it in future)
- (d) make – for testing and installing the model
- (e) Python?
- (f) gambas3-gb-xml-xslt – needed for xsltproc
- (g) Bc
- (h) Anything else you fancy?
- (i) Then click next for installation to happen

If you forget something, you can simply rerun the install executable and you can select new packages and update old ones (NB this is the only way it updates, so you have to do this occasionally anyway if you want any updates!)

NB-2 – eagle-eyed readers may have noticed we didn't mention **netCDF** here. There's a good (and annoying) reason for this, to be revealed shortly

NB-3: you may later find **Cygwin** claiming some really core commands (like rm) is missing, so cgenie won't compile. If this happens, uninstall and then reinstall **Cygwin** from scratch...

4. **Click Finish when Setup is done**, adding system icons if you want them
5. **Launch Cygwin:** Go find Cygwin Terminal (either from an icon or the Start bar) and launch it – you should get a bash-like terminal popping up ready to go
6. **Download Netcdf:** This is the other annoying step, in that netcdf should already be available as a library via Cygwin but for irritating reasons (to do with naming, library locations, and compilers) isn't properly recognised by **muffin** without some tiresome edits. So it's a lot easier just to download it yourself, install it exactly where you want it, and tell **muffin** where it is. Inelegant, but effective. If you'd prefer to make Cygwin netcdf work instead, then feel free to have a go!
 - (a) Download the netcdf4 library from the **muffin** website⁹ and save it somewhere handy like in your brand new **Cygwin** directory user area (something like C:64) where you can find it using the **Cygwin** terminal
 - (b) In **Cygwin** Terminal, navigate to where you downloaded the library to (using cd and ls commands) and then untar, configure, and install the libraries with the following code:


```
tar -xzf netcdf-4.0.tar.gz
cd netcdf-4.0
./configure --disable-shared --prefix=${HOME} FC=gfortran CC=gcc CXX=g++
# N.B. no new line here. This will also take a while...
```

⁹<http://www.seao2.info//cgenie/software/netcdf-4.0.tar.gz>

```

make check
#this checks netCDF. All the tests should pass. If not, then try again...
make install
#this installs netCDF. It should finish with a congratulations. If not, then try again...
The key thing here is making sure we use *exactly* the same compilers that installing
cGENIE will use to make sure it all matches up. Also note that this is an older version
of netcdf with both c and fortran libraries combined – this is to make life simpler here,
and still works fine

```

7. Download **muffin**: In **Cygwin** Terminal navigate back to your Home directory (where you started) and run the following code:

```

cd ..
git clone https://github.com/derpycode/cgenie.muffin.git

```

This clones the entire current cGENIE.muffin repository from github to your computer via Cygwin. Magical!

8. Configure cGENIE settings: Before installing, we need to edit a bit of code to tell cGENIE where we put netcdf. There are 2 files to edit:

- (a) Type: cd cgenie.muffin/genie-main to go to the main **muffin** directory
- (b) Type: nano user.mak in your **Cygwin** terminal to edit user.mak – at the end of the file, the **netCDF** path will need to be changed because we've put netcdf somewhere different (your home user area) to what it's expecting. Change:

```
NETCDF_DIR=/usr/local
```

to:

```
NETCDF_DIR=$(HOME)
```

and make sure this line is the same:

```
NETCDF_NAME=netcdf
```

To do so, move to the line with your keyboard arrow keys, type it in, and save on exit (in nano, press ‘Ctrl-X’ to exit and then ‘y’ to save changes or ‘n’ to not, then hit enter)

- (c) Now type: nano makefile.arc in your **Cygwin** terminal to edit makefile.arc – towards the end of the file, under the heading:

```
# === NetCDF paths ===
```

uncomment the line under (i.e. remove the `#` in front):

```
### FOR COMBINED C+FORTRAN NETCDF LIBRARIES #####
```

and comment the 2 lines under (i.e. add a `#` in front):

```
### FOR SEPERATE C AND FORTRAN NETCDF ###
```

so as to select combined, rather than separate, netCDF libraries. Save and exit.

9. **Test muffin**: now we need to check **muffin** is working.

- (a) First, make sure you're still in the directory cgenie.muffin/genie-main (type ls to find out). Then enter:

```
make testbiogem
```

After a whole bunch of text happens and time passes, you should get:

```
**TEST OK**
```

If not, try:

```
make cleanall
```

and then try again. If that doesn't work, see the full **muffin** manual FAQs for pointers, or check the earlier steps with **Cygwin** packages, **netCDF**, and **muffin** configuration worked properly

- (b) To see if it works for a proper model run, try out the following code to run a real experiment¹⁰:

```
./runmuffin.sh cgenie.eb_go_gs_ac_bg.worbe2.BASE LABS LAB_0.EXAMPLE 10
```

with the command structure following:

```
./runmuffin.sh #1 #2 #3 #4 (#5)
#1 : The base config. Here it is: cgenie.eb_go_gs_ac_bg.worbe2.BASE
#2 : The user config directory. Here it is: LABS
#3 : The user config file (i.e. the experiment name). Here it is: LAB_0.EXAMPLE.
#4 : Run the experiment for X years. Here it is: 10
#5 : The (optional) restart file. Here there is no restart, so no 5th parameter passed
```

You should see code compiling, and then after a while some results start appearing. After 10 model years (a few minutes), the model should save and finish, and you can find the outputs in `cgenie_outputs` in your Home directory.

10. **Success!** If you've made it this far, congratulations, you now having **muffin** running on your Windows machine! Remember that big runs will take a long time though, so beyond tinkering and short runs be prepared to leave your computer running for a lonnnng time. But on the upside, no queuing required!

¹⁰no new line

20.9 HOW-TO ... diagnose how the model works

Add a water mass age tracer

Water masses (and hence something of ocean circulation) can be tagged with a color (dye) tracer. However, on its own, this can tell you nothing about water mass age. A second color tracer can be added, however, and configured in such a way that by analysing the ratio of the two tracers, water mass age (time since a parcel of water last saw the surface ocean on average).

First off, you are going to need a *base-config* that defines both color tracers. An example of this (but with no biology or carbon cycle and a modern configuration) is:

```
cgenie.eb_go_gs_ac_bg.worjh2.rb
```

Obviously, this can be adapted or the 2 lines selecting the 'red' and 'blue' color tracers, copied over into a different *base-config* (but remembering that the total number of ocean tracers selected then increases by 2).

The way this is going to work is:

1. A restoring of red dye is applied evenly to the entire ocean surface. By itself, this will simply result in the ocean progressively filling up with dye until equal to the surface concentration and with no time information.
2. So a blue dye is also injected. The concentration of this is also restored at the surface. However, the surface concentration of the blue dye is scaled such that it reflects age in the model experiment. This counts 'down', such that at the start of the experiment the dye is at its highest concentration and hence representing the greatest amount of time (age). As the run progressive and time runs towards zero, so does the dye flux. i.e. for an experiment running for 10,000 years, the concentration of blue at the surface linearly declines from 10,000 to 0.

Or alternatively:

3. So far, even with the blue dye reflecting 'time', remote parts of the ocean will not have received much dye, so even though the water should be 'old' and the surface concentration high, the concentration and hence 'age' in the deep ocean will still be low. So the red dye is used to normalize for the dispersion and dilution of the blue dye.

Water mass age tracing can be configured quite simply, and with the tracer concentration manipulations carried out automatically, as follows. As *base-config* file with both red and blue color tracers defined is still required, but rather than have to define and use a set of *forcings* (and associated forcing configuration), a single parameter is added in the *user-config* file:

```
bg_ctrl_force_ocn_age=.true.
```

This will automatically create the age tracer and additional explicitly output (in netCDF) both the total age of a water parcel, as well as the age relative to the surface (ventilation age).

In this methodology, in the netCDF output, the concentration ratio of blue/red, should be 'age' – the mean time that a parcel of water was last at the surface.

In comparison, in the original (and still valid way of implementing water mass age tracing), 2 surface ocean *forcings* need be specified in order to create the combined age tracer. An example forcing is provided: *forcing*: pyyyyz.Rcolr_Rcolb, which is when configured, for a 10,000 year run in this example, by adding the following 2 parameter settings in the *user-config*:

```
bg_par_ocn_force_scale_val_49=10000.0
bg_par_ocn_force_scale_time_49=10000.0
```

(If the experiment duration is longer than 10,000 years, the parameter values need be adjusted accordingly.)

Example *user-config* files for both approaches are provided:

```
EXAMPLE.worjh2.NONE_age.SPIN
EXAMPLE.worjh2.NONE_colage.SPIN
```

Note that the automatic approach (EXAMPLE.worjh2.NONE_colage.SPIN) will handle experiments started from a *re-start* (but not for the manual approach). The only advantage to the manual approach (EXAMPLE.worjh2.NONE_age.SPIN), which is provided for backwards code/experiment compatibility, is that it is possible to specify a surface age for a specific region, e.g. North Atlantic, meaning that the ventilation age is the time since a parcel of water last saw the North Atlantic rather than anywhere at the surface (as in the automatic approach).

Include 'preformed' tracers

Preformed tracers are selected via 2 modifications:

1. By setting the parameter:
`bg_ctrl_bio_preformed=.true.`
2. By including sufficient/appropriate color tracers in the *base-config*.

A total of 10 generic color tracers are defined and can be selected (in the *base-config*), from io_col0 (tracer number #57) through io_col9 (tracer number #66), and with the preformed tracer parameter set, represent:

Ocean tracer number	Mnemonic	Represents the preformed version of tracer:
57	io_col0	DIC
58	io_col1	ALK
59	io_col2	O ₂
60	io_col3	PO ₄
61	io_col4	NO ₃
62	io_col5	Ca
63	io_col6	SiO ₂
64	io_col7	δ ¹³ C of DIC
65	io_col8	(NOT CURRENTLY USED)
66	io_col9	(NOT CURRENTLY USED)

(Obviously, you need to have the respective ocean tracer selected.)

One can have a generic *base-config*, with all 10 possible color tracers selected, and then only for the selected corresponding tracers will a preformed tracer be simulated and output generated. Or one can just select the specific color tracer corresponding to a selected biogeochem tracer.

Diagnose a Transport Matrix

The physical circulation in **muffin** can be diagnosed as a Transport Matrix: a representation of the steady-state average circulation as a sparse matrix (see *Khatiwala et al.* [2005] Accelerated

simulation of passive tracers in ocean circulation models. Ocean Modelling. 9 (1), pp. 51 - 69). There are a number of advantages of using a transport matrix such as finding preformed tracer distributions and using it as a surrogate circulation for a steady-state biogeochemical model (see Khatiwala [2007] – A computational framework for simulation of biogeochemical tracers in the ocean. Global Biogeochemical Cycles. 21 (3), GB3001).

At its most simplest, the method sets a passive tracer (i.e., affected by ocean circulation only) to $1 \mu\text{mol kg}^{-1}$ in a single grid-box, leaving every other grid-box as $0 \mu\text{mol kg}^{-1}$. The model is then integrated for a single timestep. The whole grid is then vectorised, i.e., grid-boxes are stacked on top of each other, and this forms one column of the matrix. This process is repeated for the next grid-box along forming the second column. The columns of the matrix represent the distribution of the tracer from each grid-box individually after one timestep. To diagnose a matrix within GENIE, a few extra steps are taken: 6 tracers are used in a gridded pattern to diagnose one depth level simultaneously; each depth level is diagnosed and averaged in a year in turn as the number of grid-boxes affected by convection is not predictable.

Diagnosing within a GENIE run

A full matrix can be diagnosed within a single run. To do so, first add the following six "colour" tracers selected to the *base-config* file:

```
gm_ocn_select_57=.true.
gm_ocn_select_58=.true.
gm_ocn_select_59=.true.
gm_ocn_select_60=.true.
gm_ocn_select_61=.true.
gm_ocn_select_62=.true.
```

(and remembering to increase the total number of selected ocean tracers). Then in the *user-config* file, set the following parameters values in order to diagnose a transport matrix:

1. By ensuring preformed tracers are not implemented:
`bg_ctrl_bio_preformed=.false.`
2. Select the transport matrix flag.

```
bg_ctrl_data_diagnose_TM=.true.
```

3. Select the year to start diagnosing the matrix. The matrix requires the same number of years to diagnose as there are depth levels in the configuration, e.g., 16 years with a 16-level configuration.

```
bg_par_data_TM_start=9980
```

4. Select the averaging interval. Setting to 1 will give one annual average matrix, 4 will give four seasonally averaged matrices and 12 will give monthly averaged matrices.

```
bg_par_data_TM_avg_n=1
```

There are also a few other things to consider. The model will try and warn you...

- The "colour" tracers are used for other purposes such as preformed tracers. Diagnosing preformed tracers at the same time probably isn't a good idea.

```
bg_ctrl_bio_preformed=.false.
```

- If you want to diagnose seasonal or monthly matrices, ensure you have seasonal variability otherwise you get will get a number of identical matrices:

```
ea_dosc=.true.
go_dosc=.true.
gs_dosc=.true.
```

- If you want to diagnose seasonal or monthly matrices, you may want to save other output at the same intervals by setting the following (this is not the same as the matrix averaging parameter! Set to 24 for seasonal or 8 for monthly):

```
bg_par_data_save_slice_n=24
```

Lastly, **biogem** needs to run with a 1:1 time-step ratio with **goldstein**. You can either edit the `runmuffin.sh` script yourself or better, use the `runmuffin.TM.sh` script which sets this ratio to 1 automatically.

The transport matrix data will be written to a netCDF file named `transport_matrix_COO.nc` in the biogem output (COO stands for the specific format of the sparse matrix). See the next sections on what to do with it...

Loading the Transport Matrix in MATLAB

To load the Transport Matrix into MATLAB, copy `transport_matrix_COO.nc` to a directory MATLAB can see. Use the `load_genie_matrix.m` script from www.github.com/derpycode/muffindata/Transport_Matrix to read the file and create the matrix in MATLAB and save it the matrices to a file naed whatever you specified. A matrix is conventionally called `A`. If there are multiple matrices, they are saved under one file as `A1, A2` etc...

```
load_genie_matrix ( 'transport_matrix_COO.nc' , 'output_filename')
```

The script also saves a file called `matrix_vars.mat` containing metadata:

1. `v_index` - a structure array with 4 vectors containing the indices for longitude (`i`), latitude (`j`) and depth (`k` with 16 as surface, `rk` with 1 as surface). These are required to convert between a 3-D grid to vector formats (see below).
2. `nb` - the number of wet grid-boxes
3. `Ib` and `Ii` - indices containing the location of the surface and interior ocean grid-boxes in the matrix

You should perform a basic check on the transport matrices before using them to check there were no issues in diagnosing them. Each row of the matrix must sum to 1.0 to ensure mass conservation. In practice, due to the way the matrix is averaged over a single run, this is never exact but should be close within several decimal places. Running `sum(A,2)` will find the sum of each row. `sum(sum(A,2))` should also equal the total number of boxes `nb`. Hopefully, it should be obvious that the circulation should be as close to steady state as possible in your run!

Useful MATLAB Functions

Additional MATLAB functions to help process and use transport matrices can be found at www.github.com/derpycode/muffindata/Transport_Matrix.

1. `v2f.m` and `f2v.m` - convert between vector and 3-D grid. Takes the vector indices in `v_index` as arguments.
2. `read_genie_netcdf.m` - reads one or multiple variables from a genie netCDF output file into your MATLAB workspace. It will read in a 3-D field or convert to a vector format.
3. `split_transport_matrix.m` - splits a transport matrix into a boundary (surface grid-boxes only) and interior matrices, needed when solving for steady state solutions. The function takes the `Ib` variable as an argument.

20.10 HOW-TO ... screw with the climate system

Set/un-set seasonal insolation forcing

Seasonal insolation forcing of the **EMBM**, **GOLDSTEIN** ocean, and sea-ice model, are set by the following parameters¹¹:

```
ea_dosc=.true.
go_dosc=.true.
gs_dosc=.true.
```

and are `.true.` by default. To set an annual average insolation forcing with no seasonality, simply set these to `.false..`

Implement 'brine-rejection'

cGENIE.muffin has the capability to include this effect (at least crudely) and similarly to *Bouttes et al.* [2010]. For this, three namelist parameter values need to be set:

```
bg_ctrl_force_GOLDSTEInTS=.TRUE.
bg_par_misc_brinerejection_frac=0.1
bg_par_misc_brinerejection_jmax=9
```

The first, simply allows the **BIOGEM** biogeochem. module to directly influence ocean circulation. The second is the fraction of salt, rejected during sea-ice formation (e.g., see *Bouttes et al.* [2010]) that is transferred directly to the bottom-most (underlying) ocean cell in the model. The third sets a latitude limit (counted in cells) to the effect – on a 18×18 grid, a value of 9 will restrict brine rejection to the southern hemisphere (and hence Southern Ocean); a value of 18 will allow it to take place in the North Atlantic as well. (Note that in e.g., *Bouttes et al.* [2010], the effect of brine-rejection is considered only in the Southern Ocean.)

There is also option for treating all tracers the same way (including DIC ALK, PO4, etc.) and translocate biogeochem. tracers proportionally to salinity.

¹¹These are typically set in the *base-config* if needed (i.e. different from default).

20.11 HOW-TO ... configure ocean geochemistry

Modify the ocean inventory of a tracer of a *re-started* experiment

There are three different ways in which for a closed system, the inventory of a tracer can be modified:

1. Add a flux forcing, to the ocean surface or the ocean as a whole. The tracer change is then the total global flux times the duration of the forcing. Note that the forcing can be positive or negative (to effect a decrease in the tracer inventory)
2. There is a parameter that add to or subtract from, the tracer inventory at the very beginning of an experiment (and assuming it is running on from a *re-start*). The parameter name is `bg_ocn_dinit_nn`, where nn is the tracer *number* (which can be found in the parameter list table PDF, or by inspection of the file `tracer_define.ocn` (for ocean, dissolved tracers) in the directory `cgenie.muffin/genie-main/data/input`. For example:

`bg_ocn_dinit_8=1.0E-6`

will add $1 \mu\text{M kg}^{-1}$ of PO_4 (#8 is the tracer number of dissolved phosphate), uniformly to the ocean. Note that a negative value will result in the subtraction of a uniform concentration form every grip cell in the ocean (meaning that care has to be taken to ensure that negative numbers do not appear following subtraction).

3. There is a variant to the concentration adjusting parameter that is enabled by setting the parameter

`bg_ctrl_ocn_dinit` to `false` (it is `true` by default). `bg_ocn_dinit_nn` now acts as a scaling factor that is applied to the tracer concentration field. The new concentration field is equal to the old concentration field (from the *re-start*), times $(1.0 + \text{bg_ocn_dinit_nn})$, e.g.:

`bg_ctrl_ocn_dinit=.false.`

`bg_ocn_dinit_8=0.5`

will result in a 50% increase in the concentration of dissolved phosphate everywhere in the ocean (and a value of 1.0 doubles concentrations). Conversely, a value less than one will result in a proportional reduction everywhere, e.g.:

`bg_ctrl_ocn_dinit=.false.`

`bg_ocn_dinit_8=-0.2`

generates a 20% decrease everywhere.

Obviously, if the experiment is not being run from a re-start, or is being run from a re-start which does not include the particular tracer, then the initial value of the trace can be set. The parameter name is `bg_ocn_init_nn`, where nn is the tracer *number*.

Determine the CH₄ flux required to achieve a particular atmospheric pCH₄ value

Unlike the concentration of CO₂ in the atmosphere, which if restored to a chosen value during a *spin-up* experiment, will remain at that value in a *continuation* experiment (if no other perturbation of the carbon cycle or CO₂ emissions have been prescribed), CH₄ in the atmosphere decays with a lifetime of ca. 8 years (with a small fraction dissolving in ocean surface waters and being oxidized in the ocean). Hence, atmospheric CH₄ *restored* to a particular value in a spin-up, requires that restoring to be maintained in any *continuation* experiment or CH₄ will quickly decay to zero. However, doing this (*restoring* CH₄ concentrations), prevents the effect of CH₄ emissions on being assessed (as the atmospheric composition is being held constant).

An alternative would be carry out the *spin-up* experiment with no *restoring* of atmospheric CH₄ (or

restoring to zero), and then run the *continuation* experiment with no CH₄ *restoring*. This would enable e.g. CH₄ emissions experiments to be carried out and the change in atmospheric CH₄ in response to be simulated. The problem here is that the lifetime of CH₄ in the atmosphere scales with the CH₄ concentration. So in starting with no CH₄ in the atmosphere, the CH₄ lifetime is relatively short, and the response to CH₄ emissions will be underestimated.

What is in effect 'missing' are the (natural) sources of CH₄ to the atmosphere such as wetlands, which at steady state, provide a CH₄ flux that balances the oxidation rate of CH₄ in the atmosphere (and ocean). cGENIE has a *parameter* for this: ac_par_atm_wetlands_FCH4 (mol yr⁻¹) (with the isotopic composition of this source set by: ac_par_atm_wetlands_FCH4_d13C). All that then remains is to determine the flux of CH₄ that balances the rate of oxidative loss for the desired atmospheric CH₄ concentration. To do this:

1. Carry out a *spin-up* with atmospheric CH₄ *restored* to the desired concentration.¹²
2. Determine the total loss rate of CH₄ (including both atmospheric oxidation and invasion (and subsequent oxidation) into the ocean) – this is recorded in the *time-series* results file: biogem_series_focnatm_pCH4.res¹³.
3. Set the *parameter* ac_par_atm_wetlands_FCH4 equal to this value.

An example of a spin-up in which a prescribed ('wetland') flux of CH₄ to the atmosphere is set, is described in:

cGENIE.Examples – *spin-up* example EXAMPLE_p0055c_P04_CH4_SPIN2

Determine the atmospheric radiocarbon flux required to achieve a steady state ¹⁴C system

First, you need to spin the system with the atmosphere restored to the e.g. pre-industrial Δ¹⁴C value desired. There are two ways then to diagnose the equivalent ¹⁴C flux:

1. Sum the total mol inventory of Δ¹⁴C in the ocean (as DIC) and atmosphere (CO₂), both of which can be found in the relevant time-series output. The radiocarbon decay rate multiplied by the (steady state) inventory then gives the total decay, which of course is equal to the cosmogenic input flux needed to maintain steady state.
2. Rather simpler, enabled by some recent output code changes, is simply to read off the reported atmospheric flux forcing that is being applied to restore the atmosphere Δ¹⁴C value. The relevant time-series output file is:

biogem_series_diag_misc_specified_forcing_pCO2_14C.res

Once the balancing ¹⁴C flux value has been obtained, the namelist parameter controlling the rate of cosmogenic ¹⁴C production (by default, zero): ac_par_atm_F14C

¹²For an example: see experiment EXAMPLE_p0055c_P04_CH4_SPIN described in *cGENIE.Examples*.

¹³Second column (the value in units of mol yr⁻¹)

20.12 HOW-TO ... biology

Configure an abiotic ocean

Biological productivity in the ocean can be completely turned off to create an abiotic ocean (why you would want to do this is another matter ... perhaps analyzing the solubility pump or a 'deep-time' and prior to significant marine life study ... (?)). The biological option is set by the *parameter*:

```
bg_par_bio_prodopt
```

which by default takes a value of "1N1T_P04MM" which selects the scheme described in *Ridgwell et al.* [2007a]. To have no biological production in the ocean, add the following line to the end of the *user-config* file¹⁴:

```
bg_par_bio_prodopt="NONE"
```

With this set, you do not have to specify any biological production or remineralization *namelist parameter* values in the *user-config* file.

Prescribe the CaCO₃:POC export ratio

In the default¹⁵ 'biological' scheme in GENIE the CaCO₃:POC export ratio from the surface ocean in BIOGEM is parameterized as a power law function of the degree of ambient over-saturation w.r.t. calcite [*Ridgwell et al.*, 2007a,b]. The calculated CaCO₃:POC ratio will vary therefore both spatially, particularly w.r.t. latitude (and temperature), as well as in time, if the surface ocean saturation state changes. The latter can arise from climatic (temperature) or circulation changes, or through a change in the DIC and/or ALK inventory of the ocean (such as resulting from emissions of fossil fuel CO₂) or the re-partitioning of these species vertically within the ocean (e.g., as a result of any change in the strength of the biological pump).

There may be situations in which it is advisable to hold the CaCO₃:POC export ratio invariant. For instance, considering the current very considerable uncertainties in the impacts of ocean acidification on marine calcifiers [*Ridgwell et al.*, 2007a] the safest assumption is arguably to exclude any acidification impact on calcification and carbonate export. Specifying a spatially uniform value of the CaCO₃:POC ratio ratio (e.g. 0.25 or 0.3) also allows comparison with the results of early carbon cycle model studies. For deeper-time geological studies where little about marine carbonate production may be known *a priori*, a spatially uniform value represents the simplest possible assumption (e.g., *Panchuk et al.* [2008]).

BIOGEM can be told to use a prescribed (spatially and temporally invariant) 2D field of CaCO₃:POC export rain ratios (instead of calculating these internally as a function of ocean chemistry) by setting the 'Replace internal CaCO₃:POC export rain ratio?' *namelist parameter* flag to .true.:

```
bg_ctrl_force_CaCO3toPOCrainratio=.true.
```

You must also then provide a 2D data field that specifies the value of the rain ratio at each and every surface ocean grid point. The filename of this field is set by default to:

```
bg_par_CaCO3toPOCrainratio_file="CaCO3toPOCrainratio.dat"
```

and the file must be located in the 'BIOGEM data input directory'¹⁶, which by default is:

```
bg_par_indir_name="$RUNTIME_ROOT/genie-biogem/data/input"
```

This 2-D field must be in the form of an ASCII file with space (or tab) separated values arranged in rows and columns of latitude and longitude. The format of the file must follow the GOLDSTEIN

¹⁴Or edit the existing line under the section '-- BIOLOGICAL NEW PRODUCTION --'

¹⁵The default biological scheme is given by: `bg_par_bio_prodopt='1N1T_P04MM'`.

¹⁶\$RUNTIME_ROOT being equal to ~/genie.

ocean grid with the first line being the most Northerly row, and the last line the most Southerly row of grid points. Data along a row is from West to East. The latitude of the first column of values must be consistent with the defined starting latitude of the model grid, which is specified by the namelist parameter `gm_par_grid_lon_offset`¹⁷. Examples are given in the code repository¹⁸.

If you are using a uniform value, it is an easy enough job to create a 36×36 array of the value you so desire¹⁹.

If you want to hold a previously-calculated (spatially variable) CaCO₃:POC field constant, then the easiest way to achieve this is to copy the information contained in the *time-slice* results field: `misc_sur_rCaCO3toPOC` in the results netCDF file `fields.biogem_2d.nc`²⁰. Because this is a 3D data field ($36 \times 36 \times 8$), carefully highlight just the surface ocean (2D) distribution (e.g., from the Panoply viewer) or extract from the netCDF file by some other means, and then copy and paste into:

`CaCO3toPOCrainratio.dat` (or whatever you have specified the filename as). When copying Panoply data, 'NaN's should be replaced by values of zero. Take care that the final (steady-state) time-slice is being copied and not the first (un-spunup) one ...

TIP: In order to quantify the importance of calcification feedbacks with CO₂ and climate, two model integrations are required: one with the CaCO₃:POC ratio held constant and the other with it allowed to vary, thereby allowing the effect of a changing CaCO₃:POC ratio on the system to be elucidated.

Prescribe biological export production

Two possibilities:

1. **Via a full prescription of all particulate fluxes in surface ocean**

Create a full set of particulate (sediment tracer) flux forcings fields for the surface ocean, one for each biologically-related sediment tracer selected in the model, including isotopes (and trace metals). Everything except for the surface layer can be left as a zero (0.0) in the two 3D spatial fields required for each tracer.

You must also create a set of dissolved (ocean) tracer flux forcings fields for the surface ocean, one for each dissolved tracer associated with the particulates and selected in the model (including isotopes etc). The dissolved tracer flux fields must be created so as to exactly cancel out the particulate fields to conserve mass. For most tracers this is trivial, i.e., the fields for P in particulate organic matter (`sed_POP`) need be associated with fields for dissolved PO₄ (`ocn_PO4`) which will simply be equal in magnitude but opposite in sign to POP. Complications start to arise for CaCO₃ and there is also the question of alkalinity changes associated with organic matter creation/destruction (via changes in NO₃), so this method, whilst the most flexible, is not without its complications and degree of tediousness (i.e. would not recommend).

2. **By just prescribing just the POC flux**

An alternative has been provided enabling a full biological productivity in the surface ocean, but controlled by prescribing just the particulate organic carbon export flux. This 'biological' scheme is selected with:

¹⁷-260E by default

¹⁸e.g., `~/genie/genie-biogem/data/input/CaCO3toPOCrainratio_worbe2_preindustrial.dat`

¹⁹It doesn't matter if you specify a value over land because only values associated with wet cells will be acted on.

²⁰You must have the 'miscellaneous properties' time-slice save flag set to:

`bg_ctrl_data_save_slice_misc=.true.` (the default) for this field to be saved.

```
bg_par_bio_prodopt="bio_POCflux"
```

What happens in practice is that the POC flux is used to calculate the equivalent PO₄ change in the surface ocean, and then this is passed to the biological scheme and export production calculated 'as usual'. (The POC flux forcing is set to zero once the associated PO₄ (uptake) flux has been calculated.)

A particulate (sediment tracer) flux forcing for POC in the surface ocean still has to be defined and selected, but no other *forcings* (including the associated removed dissolved tracers) are required. An example forcing configuration is given in worjh2.FPOC_Caoetal2009 (and selected by: `bg_par_forcing_name="worjh2.FPOC_Caoetal2009"`) An example *user-config*: EXAMPLE.worjh2.Caoetal2009_FPOC illustrating this is provided.

NOTE: Take care with dissolved organic matter (DOM) production, as the specified POC flux is automatically increased to take into account DOM production. i.e. if 50% of export is specified to be partitioned into dissolved rather than particulate organic matter export, whatever is specified in the POC export forcing would be internally doubled, before being partitioned into POM and DOM. **NOTE:** Also take care with the units of the flux *forcing* to the surface layer in the ocean, which are in mol yr⁻¹. The main particulate flux output is in units of mol m⁻² yr⁻¹, but since cGENIE is invariably run on a equal area grid it is not difficult to convert export production densities to mol yr⁻¹ (you either need to divide the area of the Earth's surface by the number of grid points, or save the ocean grid information – see netCDF save options in the User Manual). Alternatively, as of revision r8825, cGENIE saves the primary particulate flux fields also in units of mol yr⁻¹ (assuming you have the biological or full netCDF save options selected – see netCDF save options in the User Manual).

Be aware that if there is insufficient PO₄ to support the require POC flux, the entire POC flux will still be created, meaning that you may end up with regions of negative nutrient concentration.

Include a R-DOM cycle in the ocean

The parameter: `bg_ctrl_bio_remin_RDOM_photolysis` determines whether RDOM degradation is restricted to the surface layer and occurs only by/associated with photolysis. It can be `.true.` or `.false.` and by default is set to:

```
bg_ctrl_bio_remin_RDOM_photolysis=.false.
```

When set `.true.`, RDOM degradation is set to zero everywhere in the ocean except the surface layer. Here, the lifetime (parameter: `bg_par_bio_remin_RDOMlifetime`) is modified in *inverse* proportion to the solar insolation integrated over the surface layer. (There is a field in the 2D netCDF of solar insolation at the ocean surface, and the average over the surface layer is approx 1/4 of this.). i.e., in lower latitude and higher insolation regions, the lifetime is shorter than specified by `bg_par_bio_remin_RDOMlifetime` (and by approx a factor of 1/4 of the solar insolation in W m⁻²).

Include a Cd cycle in the ocean

In order to run cGENIE with ocean cadmium cycle, the following *base config*: `cgenie_eb_go_gs_ac_bg_itfcld_161_JH_BASEFeCd` is provided.

A typical experiment command line, using the *user config* file: EXAMPLE_worjh2_P04Fe_Cd_SPIN (also provided under SVN), would look like:

```
./runCCgenie.sh cgenie_eb_go_gs_ac_bg_itfcld_161_JH_FeCdBASE /  
EXAMPLE_worjh2_FeCd_SPIN 11
```

To submit this job to the cluster (from \$HOME):

```
qsub -q kitten.q -j y -o cgenie_log -S
/bin/bash subcgenie.sh cgenie_eb_go_gs_ac_bg_itfcld_16l_JH_BASEFeCd /
EXAMPLE_worjh2_P04Fe_Cd_SPIN 10001
```

Include an iodine cycle in the ocean

In order to run cGENIE with a marine iodine cycle, one of the following *base-configs* is needed:

```
cgenie.eb_go_gs_ac_bg.worjh2.BASEI
cgenie.eb_go_gs_ac_bg.worjh2.BASEI
cgenie.eb_go_gs_ac_bg.worbe2.BASESI
```

Of these, recommended is one of the 16-level ocean worjh2 configurations²¹ as the oxygen minimum zones in the 8-level ocean worbe2 configuration are much more poorly developed (see Ridgwell *et al.* [2007a]).

Example configurations of several different levels of complexity of iodine cycling are given in the follow sections.

Basic ('abiotic') iodine cycle

This section outlines the most trivial possible configuraion of the marine iodine cycle, in which the only processes are:

1. Reduction if IO₃⁻ to I⁻ under dysoxic conditions
2. Re-oxidation of I⁻ to IO₃⁻

An example experiment configuration *user-config* – EXAMPLE.worjh2.P04Ibasic.SPIN – is given. In this, the available parameters (listed under the heading '# -- MISC --' and '### IODINE CYCLE CONTROLS ###') controlling the iodine cycle are:

- # set no biological IO₃ uptake

For simplicity, this experiment configuration sets the iodate update in organic matter associated with biological production at the ocean surface, to zero. The parameter bg_par_bio_red_POC_POI specifies the ratio of I to C in new production (the cellular quotient) and is set to zero.

- # select basic reduction and oxidation options

There are various alternative options for how IO₃ is reduced in dysoxic conditions. The simplest parameterization is specified here;

bg_opt_bio_remin_reduce_I03toI='threshold',

in which a threshold of dissolved oxygen is prescribed. In any regions (i.e. model grid boxes) of the ocean in which dissolved oxygen concentrations fall are below this, IO₃⁻ is completely reduced to I⁻.

Alternative options also exist for how the re-oxidation of I⁻ occurs. In the simple parameterization specified here;

bg_opt_bio_remin_oxidize_ItoI03='lifetime',

a fixed lifetime of I⁻ in the ocean is prescribed. Oxidation proceeds at this rate regardless of the oxygenation state of the ocean, but as long as sufficient oxygen to accomplish the reaction $2I^- + 3O_2 \rightarrow 2IO_3^-$ exists.

²¹The difference between the two worjh2 configurations is that one (with the 'S' in 'BASESI' also includes a sulphur cycle.

- # set [O₂] threshold (mol kg⁻¹) for (complete) reduction of I⁻.

The parameter bg_par_bio_remin_cO2_I03toI sets the dissolved oxygen concentration threshold (mol kg⁻¹, below which IO₃⁻ will be reduced).

- #set I lifetime (yrs)

Finally, the parameter bg_par_bio_remin_Ilifetime then sets the lifetime of I⁻ in years.

Model output is saved in the 'normal way' (refer to the User Manual) and amongst the ocean tracers ('ocn_*'), are the tracers of dissolved iodide and dissolved iodate (in units of mol kg⁻¹). Model output can also be contrasted with observed data re-gridded to the cGENIE (worjh2) grid. A typical command-line launching of a model experiment (10000 years integration in this case) would be:

```
./runmuffin.sh cgenie.eb_go_gs_ac_bg.worjh2.BASEI /
EXAMPLE.worjh2.P04Ibasic.SPIN 10000
```

'Biotic' component to the iodine cycle

This section outlines the next component of the marine iodine cycle, involving phytoplankton, with the two processes:

1. uptake of IO₃⁻
2. remineralization/release of I⁻

The way this cycling in the model works is as follows: First, as plankton biomass is created, alongside C and P (and if selected N, Fe and other trace elements), IO₃⁻ is taken up by the cell, in a specified proportion to carbon (see below). Of this biomass, a proportion is assumed to be exported in particulate organic matter (POM) form beneath the euphotic zone, with the remainder converted to labile dissolved organic matter (DOM) (see: Ridgwell *et al.* [2007a] for details). When either POM or DOM is remineralized and elemental constituents released, rather than returning IO₃⁻ back to solution, it is assumed that the dissolved iodine is in the form of I⁻. (This also requires an accounting of release of O₂, as IO₃⁻ is taken up by the cell and assumed to be instantaneously internally reduced.) The result is: (i) a progressive transformation IO₃⁻ → I⁻ in the surface ocean as DOM is continually created and destroyed, and (ii) a 'nutrient-like' enrichment of I⁻ in the sub-surface as a consequence of the remineralization of POM.

An example experiment configuration *user-config* – EXAMPLE.worjh2.P04Ibio.SPIN – is given. In this, the available parameters (listed under the heading '# -- MISC --' and '### IODINE CYCLE CONTROLS ###') controlling the iodine cycle are:

- # select option for no watercolumn reduction

For simplicity, this experiment configuration sets the reduction of iodate under low oxygen conditions in the water to zero (i.e. leaving the biological pump as the only source of I⁻);
bg_opt_bio_remin_reduce_I03toI='NONE'

- # set biological I03 uptake

Set a cellular I:C quotient. Here, a value of 1.0E-4 is given as an example;
bg_par_bio_red_POC_POI = 1.0E-4

- # set I lifetime (yrs)

Finally, the parameter bg_par_bio_remin_Ilifetime then sets the lifetime of I⁻ in years. (Here, for illustrative purposes, the lifetime is increased to 10 years compared to 1 year in the basic Example.)

20.13 HOW-TO ... remineralize ... differently

Overview

Implement 'instantaneous remineralization'

By default in the original version of cGENIE, biogenic particles exported out of the surface ocean layer were assumed to settle at velocity of 125md^{-1} . The parameter controlling this was: `bg_par_bio_remin_sinkingrate`. The consequence was that particles took more than a single time-step to reach the ocean floor and be fully remineralized. Hence, particles could be found at intermediate depth at the ocean at the end of any particular time-step; settling further during the next time-step (and so on to the deepest ocean layer at that grid point).

An associated parameter was defined to control the residence time of particles in any one layer for the purpose of calculating scavenging. A separate parameter was created so that the residence time and hence scavenging rate could be varied independently of the settling rate of the bulk particles. This parameter was: `bg_par_bio_remin_sinkingrate_scav`. The default value of this is also 125md^{-1} .

This all became pretty confusing ... what is 'real' settling and what the 'scav' bit is ... so a new pair of parameters was created. These are:

- `bg_par_bio_remin_sinkingrate_physical`
- `bg_par_bio_remin_sinkingrate_reaction`

The former defining the 'real' (physical) sinking speed, and the latter the effective sinking speed for the purpose of calculating the residence time in each ocean layer and hence reaction rates. Here, reaction rates include scavenging as well as opal dissolution, and organic matter remineralization driven by temperature.

By default these are set to zero, and the values of `bg_par_bio_remin_sinkingrate` and `bg_par_bio_remin_sinkingrate_scav` are used.

As an alternative to the default, of particles settling a finite distance down through the water column each time-step, the products of remineralization can be instantaneously distributed down through the water column – as if the particles settled with infinite velocity, yet still dissolved and reacted at the same rates as before, i.e. at a finite settling speed. To do this one would set, e.g.:

```
bg_par_bio_remin_sinkingrate_physical=1.0E9
bg_par_bio_remin_sinkingrate_reaction=125.0
```

At a velocity of 1.0^9md^{-1} , particles would reach the ocean floor (regardless of the depth at that grid point) in a single time-step. Remineralization is therefore calculated throughout the water column within that time-step – 'instantaneous'. The second parameter says that for the purposes of residence time and reactions, sinking should be considered to have occurred at 125md^{-1} .

Why do this? It simplifies calculation of flux mass balances, as material leaving the ocean surface is 'seen' (arriving) at the sediment surface at the same time-step. This behaviour is likely to become the **muffin** default in the future.

Alternative schemes

Implement an alternative fixed remineralization profile for POC (e.g. Martin curve)

There are several options for utilizing a fixed remineralization profile for POC, which by default is a double exponential (See: *Ridgwell et al.* [2007a]). The fixed remineralization profile scheme is set by the string parameter: `bg_par_bio_remin_fun`. By default, it has a value of 'efolding'. Currently available options are:

- *Martin1987*, which applies a globally-uniform power, set by:
`bg_par_bio_remin_martin_b`
(which by default has a value of -0.858)
- *Henson2012*, which calculates the value of b according to sea surface temperature (SST):
 $b = (0.024 * \text{SST}) - 1.06$

To user either (on their own), all organic matter should be assigned to a single phase, with the 2nd (recalcitrant) fraction set to zero:

```
bg_par_bio_remin_POC_frac2=0.0
```

Note that these parameterizations can be combined with ballasting and will act on the 'free' POC phase (i.e. the one not controlled by the ballasting parameterization).

Implement particulate organic carbon 'ballasting'

The default particulate organic carbon (POC) ocean interior remineralization scheme is based on fixed, prescribed profiles of relative POC flux to depth (e.g. see: *Ridgwell* [2001]; *Ridgwell et al.* [2007a]). A 'ballasting' control on POC transport to depth can instead be implemented by:

```
bg_ctrl_bio_remin_POC_ballast=.true.  
bg_ctrl_bio_remin_POC_fixed=.false.
```

The POC 'carrying coefficients' for CaCO₃, opal, and detrital (lithogenic) material are set by the parameters:

```
bg_par_bio_remin_ballast_kc  
bg_par_bio_remin_ballast_ko  
bg_par_bio_remin_ballast_kl
```

(for CaCO₃, opal, and lithogenics, respectively). Note that the ballast coefficient units are: g POC m⁻² yr⁻¹ (g ballast m⁻² yr⁻¹)⁻¹ (i.e. **g g-1**), which are internally converted to: mol POC m⁻² yr⁻¹ (mol ballast m⁻² yr⁻¹)⁻¹ (i.e. **mol mol-1**).

A fixed (in time), but spatially heterogeneous field can also be prescribed instead of global uniform values (akin to setting a pattern of the CaCO₃:POC export rain ratio ??). The parameters setting whether to substitute a globally-uniform value with a specified pattern are:

```
bg_ctrl_force_CaCO3ballastcoeff=.true.  
bg_ctrl_force_opalballastcoeff=.true.  
bg_ctrl_force_detballastcoeff=.true.
```

and which by default are `.false.`. The patterns of carrying coefficient are determined by files read in from cgenie/genie-biogem/data/input. The filenames are specified by:

```
bg_par_CaCO3ballastcoeff_file  
bg_par_opalballastcoeff_file  
bg_par_detballastcoeff_file
```

(again, akin to the methodology for setting the CaCO₃:POC export rain ratio (??)).

Note that ballasting is combined with an e-folding (or other) fixed profile remineralization schemes²². Ballasting is calculated with respect to the 2nd (recalcitrant) fraction of POC only. The remaining POC export is degraded by an alternative algorithm, which by default, is an e-folding decay (see previously for more and alternatives). The fraction of initial export assigned to ballasting vs. 'free' POC is calculated according to the available exported ballast flux.

²²Although in a sense, the remineralization of POC is not 'fixed' in that it does not have a predetermined profile but instead is set by the changing flux of CaCO₃, opal and lithogenic fluxes with depth, `bg_ctrl_bio_remin_POC_fixed` should still be set to `.true.` (the default).

20.14 HOW-TO ... force the system

(general)

Note: All the *forcings* described here assume a newer (simplified) methodology for prescribing *forcings*.²³ This methodology is enabled by setting:

```
bg_ctrl_force_oldformat=false.
```

This is set automatically as part of the `runmuffin.sh` shell script²⁴.

Taking the example of the ocean (dissolved tracers): flux and restoring *forcings* are defined in the *forcings* specification file: `configure_forcings_ocn.dat`. As detailed in the notes to this file, there is a flag (COLUMN #6) which sets the spatial attributes of the *forcing* as follows:

- 3 – '3D' – force the entire ocean volume uniformly
- 2 – '2D' – force the entire ocean surface uniformly
- 0 – '0D' – force a specified point
- 1 – 'SURFACE' – force with/to specified surface ocean pattern
- 2 – 'BENTHIC' – force with/to specified benthic pattern
- 3 – 'LEVEL' – force a specific layer in the ocean with/to specified pattern²⁵
- 4 – 'SURFACE+BENTHIC' – simultaneously force with/to specified (and sperate) surface and benthic pattern

The default (3) is that the forcing is applied uniformly to the entire (3D) ocean volume.

Options 3, 2, and 0, as: uniform 3D²⁶ (volume), uniform 2D (surface), and point forcing, respectively, require no additional (spatial) information and only an additional file specifying the time-dependent information for each forcing need be provided:

```
biogem_force_flux_ocn_xxx_sig.dat
```

for flux forcings, and

```
biogem_force_restore_ocn_xxx_sig.dat
```

where: `xxx` represents the mnemonic of the tracer (e.g., DIC is dissolved inorganic carbon. CH4 is methane, etc.).

Options -1 (SURFACE), -2 (BENTHIC) -3 and -4, require a 2D field to be provided, in addition to the time-dependent information for each forcing. The grids for both are the same – i.e., all 'wet' grid points (non dry land) in the model. The filename for these 2D files is of the form:

```
biogem_force_flux_ocn_xxx_SUR.dat
```

for flux forcings, and

```
biogem_force_restore_ocn_xxx_SUR.dat
```

with BEN for the equivalent BENTHIC, and LEV for LEVEL *forcing* selections.

²³For details of the 'old', fully 3D spatially-explicit forcing methodology, refer to the *cGENIE user-manual*.

²⁴(but it not the actual default *namelist parameter* setting)

²⁵For k=k_max, this duplicates the SURFACE forcing.

²⁶Note that here: '3D' does not mean a spatially explicit 3D pattern and hence the original ('old') way of specifying *forcings*, but instead: that the forcing is applied uniformly in 3D space (i.e., is in effect a volume *forcing*).

Prescribe a time-varying history of radiocarbon production in the atmosphere

The original way for doing was this was to create a CO₂ flux forcing and give its δ¹⁴C signature an extremely positive value. However, inputs and outputs of isotopic compositions in cGENIE are limited to being between -999 and +999 per mil. Prescribing a small flux of CO₂ to the atmosphere with a δ¹⁴C of +999 per mil will not give an equivalent flux as if it were pure ¹⁴C (hence needing the calculation of the equivalent total CO₂ flux with a +999 per mil signature).

There is now a simple hack (not yet extended to flux forcings of the ocean) to direct cGENIE to interpret the value specified as its δ¹⁴C forcing signature as an absolute flux (in units of mol yr⁻¹) rather than convert from a per mil notation. cGENIE knows to use this alternative if the flux of CO₂ specified in the CO₂ flux forcing is *identical* to the value given for ¹⁴C. i.e. in order to prescribe a production rate of ¹⁴C in the atmosphere sufficient to balance ocean-atmosphere carbon cycling (in the Cao *et al.* [2009] configuration of the model), instead of setting the atmospheric ¹⁴C production parameter:

```
ac_par_atm_F14C=0.387E+03
```

One would now set an identical value of 0.387E+03 in both *forcing* files:

```
biogem_force_flux_atm_pcO2_14C_sig.dat  
biogem_force_flux_atm_pcO2_sig.dat
```

(or equivalently, re-scale a unit flux forcing given in the *forcing* files using the bg_par_atm_force_scale_val_5 parameter in the example of ¹⁴C (and bg_par_atm_force_scale_val_3 for bulk CO₂)).

Selection of the required *forcings* in *configure_forcings_atm.dat*²⁷ then looks like this:

```
-START-OF-DATA-
03 f 0.1 t t F 2 01 01 '[carbon dioxide (CO2)]',
05 f 0.1 t t F 2 01 01 '[14C CO2]',
-END-OF-DATA-
```

Now that ¹⁴C production is being specified explicitly by means of a *forcing*, one can easily then implement a time-dependent change in ¹⁴C production.

Prescribe an injection of radiocarbon-dead DIC

First ... a *base-config* with 14C tracers is needed, e.g.,:

```
cgenie_eb_go_gs_ac_bg_itfcclsd_161_JH_ANTH
```

or

```
cgenie_eb_go_gs_ac_bg_itfcclsd_161_JH_ANTHFe
```

(with Fe co-limitation of marine biological productivity).

Then, in the *user-config*, an appropriate *forcing* needs to be specified, e.g.:

```
pyyyyyx_FDIC_F13DIC_F14DIC
```

and under the heading -- FORCINGS --, might look something like:

- For *forcing* selection:

```
bg_ctrl_force_oldformat=.false.  
bg_par_forcing_name="worjh2_FDIC_F13DIC_F14DIC"
```

which prescribes a forcing of DIC plus its (13C and 14C) isotopes to the ocean (somewhere or everywhere).

- Then:

```
bg_par_ocn_force_scale_val_03=0.0833e15
```

²⁷There is no need to specify a ¹³C flux (ignore any warnings at start-up). Indeed, formally, there is no ¹³C associated with ¹⁴C production from N₂. Actually ... DO NOT add a ¹³C forcing, just in case ...

sets the flux (mol yr⁻¹), which is equivalent to 1 PgC yr⁻¹.

- To scale the isotopic composition:

```
bg_par_ocn_force_scale_val_04=-60.0
bg_par_ocn_force_scale_val_05=-999.0
```

for example gives -60 per mil for 13C like methane and 14C that is pretty isotopically dead²⁸.

- By default in the *forcing*, the duration of the emission 1 year, and can be re-scaled (e.g., to 1000 years duration) by:

```
bg_par_ocn_force_scale_time_03=1000.0
bg_par_ocn_force_scale_time_04=1000.0
bg_par_ocn_force_scale_time_05=1000.0
```

- Finally – the emission location is specified by, e.g.:

```
bg_par_force_point_i=18
bg_par_force_point_j=26
bg_par_force_point_k=7
```

Alternatively, the DIC release can be made over the entire ocean floor or simply sections (or depth intervals) of the ocean floor instead of a point source.

Prescribe a spatial map of benthic tracer release

Flux *forcings* to the ocean with a variety of different spatial attributes can be specified in the *forcings* specification file: `configure_forcings_ocn.dat`. As detailed in the notes to this file, there is a flag (COLUMN #6) which sets the spatial attributes of the *forcing*:

- 3 – '3D' – force the entire ocean volume uniformly
- 2 – '2D' – force the entire ocean surface uniformly
- 0 – '0D' – force a specified point
- 1 – 'SURFACE' – force with/to specified surface ocean pattern
- 2 – 'BENTHIC' – force with/to specified benthic pattern
- 3 – 'LEVEL' – force a specific layer in the ocean with/to specified pattern²⁹
- 4 – 'SURFACE+BENTHIC' – simultaneously force with/to specified (and separate) surface and benthic pattern

Options -1 through -4 require a 2D field to be provided. The grids for both are the same – i.e., all 'wet' grid points (non dry land) in the model. Templates for these can be created as follows:

1. Open up the *BIOGEM* results file: `fields_biogem_2d.nc` (any experiment).
2. Display the variable: `grid_mask`.
3. Select the Array 1 tab (to display the actual gridded values rather than the color-coded map); highlight the grid of values and then copy-and-paste to a text editor.
4. You should have a grid of values, with a '1.0' representing ocean, and 'NaN' land. The NaNs can then be search-and-replaced to '0.0' and you have a grid valid for either the entire surface ocean or entire benthic surface.

From here: 1s can be replaced by 0s to remove unwanted locations.³⁰

²⁸Note this is on the scale of d14C not D14C

²⁹For k=k_max, this duplicates the SURFACE forcing.

³⁰This can be quite time-consuming and tedious and there is no particular short-cut :(

In the forcing configuration file, if the COLUMN #5 flag ('scale flux forcing of tracer?') is set to 't', then the flux applied at each selected location is scaled such that the total applied flux is equal to that given in the *forcing* time-signal file.³¹

Applying a geoengineering 'liming' flux to the ocean surface

[see geoengineering HOWTO]

³¹The values in the forcing map need not be all 1.0 of course.

20.15 HOW-TO ... do stuff with sediments

Spin-up the full marine carbon cycle including deep-sea sediments

By a 2-step process³²:

1. First-guess closed system *spin-up*

As of code release **r4211**, it is possible to carry out the initial spin-up, with a solute input to the ocean via rivers, but also at the same time, with the system configured closed, i.e.,:

```
bg_ctrl_force_sed_closesystem=.true.
```

The weathering flux is subtracted from ocean cells overlying the sediments to balance the global budget and ensure a closed system. This subtraction involves partitioning the total global weathering flux between each ocean floor cell with a subtraction in proportion to the estimated $CaCO_3$ preservation and burial rate. To utilize this methodology now requires that the **ROKGEM** module is used, i.e., a *base config* such as:

```
cgenie_eb_go_gs_ac_bg_sg_rg_itfcldsd_161_JH_BASE
```

A first guess for the weathering flux must now be prescribed. This could be derived from a previous closed system model experiment with no weathering flux specified (diagnosing weathering from total global $CaCO_3$ burial as described earlier), or from the literature, e.g., Ridgwell [2007] cites 20 $Tmol HCO_3^- yr^{-1}$, an equivalent $CaCO_3$ weathering rate of 10 $Tmol yr^{-1}$:

```
rg_par_weather_CaCO3=10.00E+12
```

The following *user config* file

```
EXAMPLE_worjh2_P04_S36x36_SPIN
```

can be used for the closed system spin-up.

To launch an experiment, type (all in one line; notes space separators between line items in this document format):

```
./runCCSgenie.sh cgenie_eb_go_gs_ac_bg_sg_rg_itfcldsd_161_JH_BASE /  
EXAMPLE_worjh2_P04_S36x36_SPIN 20001
```

To submit to the cluster type:

```
qsub -q kitten.q -j y -o cgenie_log -S /bin/bash subcgenie.sh  
cgenie_eb_go_gs_ac_bg_sg_rg_itfcldsd_161_JH_BASE /  
EXAMPLE_worjh2_P04_S36x36_SPIN 20001
```

20,000 years is probably about the minimum practical *spin-up* time. Primarily – you are looking for convergence in the mean wt% $CaCO_3$ value (averaged sediment composition), which is recorded in the **BIOGEM** time-series file:

```
EXAMPLE_worjh2_P04_S36x36_SPIN
```

2. Open system *spin-up*

The last stage is an open system spin-up as described previously. The prescribed weathering flux (`rg_par_weather_CaCO3`) is revised and set equal to the diagnosed global $CaCO_3$ burial rate ('Total CaCO3 pres (sediment grid)') as reported in the **SEDGEM** module results file:

`seddiag_misc_DATA_GLOBAL.res`. In addition, an *open system* must now be specified in the *user config*:

³²This is a revised methodology compared to that described in the GENIE-1 HOW-TO.

```
bg_ctrl_force_sed_closesystem=.false.
```

50,000 years is probably about the minimum practical *spin-up* time. Again – you are looking for convergence in the mean wt% $CaCO_3$ value and it is up to you to judge how long the *spin-up* needs to be.

There is still some departure of ocean Ca and ALK inventories during the revised multi-stage spin-up compared to observed (and the initialized values), but this is substantially reduced compared to the original 2-part spin-up methodology as well as to a single spin-up methodology.

TIP: Having completed the full marine carbon cycle spin-up, it is recommended that the $CaCO_3 : POC$ rain ratio is set invariant – see earlier HOW-TO. If the default $CaCO_3$ parameterization setting is retained, CO_2 -calcification feedback as described in *Ridgwell et al. [2007b]* is enabled.

NOTE: There is no climate feedback by default. To run experiments with feedback between CO_2 and climate, add:

```
ea_36=y
```

at the end of the *user-config*.

Run the sediments at higher resolution (as compared to the ocean grid)

By default (as set in the *base-config* file in `~/cgenie.muffin/genie-main/configs`) the **SEDGEM** sediment grid is configured at a resolution of 36×36 (and on an equal area grid), by:

```
SEDGEMNLONOPTS='$(DEFINE) SEDGEMNLONS=36'
SEDGEMNLATSOPTS='$(DEFINE) SEDGEMNLATS=36'
```

Several data input files are required by **SEDGEM** consistent with the specified grid:

- A mask, which specifies the sediment grid locations (if any!) at which sediment cores (see: *Ridgwell [2007]*) are to be generated at:

```
sg_par_sedcore_save_mask_name="sedgem_save_mask.36x36"
```

The example provided on SVN contains some illustrative locations set (by a '1') for cores to be generated.

- The required sediment grid topography (bathymetry):

```
sg_par_sed_topo_D="sedgem_topo_D.36x36"
```

This particular grid is derived from observed bathymetry and excludes sediment locations shallower than the surface ocean layer (of the 8-level model) as described in *Ridgwell and Hargreaves [2007]*.

As described in Ridgwell and Hargreaves [2007], **SEDGEM** can be sub-gridded to a resolution of 72×72 (equal area). The *following namelist* parameter additions are necessary to the *user-config* file:

```
SEDGEMNLONOPTS='$(DEFINE) SEDGEMNLONS=72'
SEDGEMNLATSOPTS='$(DEFINE) SEDGEMNLATS=72'
sg_par_sed_topo_D="sedgem_topo_D.72x72"
sg_par_sedcore_save_mask_name="sedgem_save_mask.72x72"
```

NOTE: Carbonate chemistry stability problems (= model crash) may occur in the 16-level configuration in conjunction with 72×72 resolution sub-gridded sediments. Who knows why?! :(

Include shallow water depositional systems

By default, the entire seafloor grid is considered as '*deepsea*' and sedimentary diagenesis options are provided accordingly. In practice, because the simple diagenesis options available, such as for CaCO₃ (e.g. *Archer* [1991]; *Ridgwell* [2001]; *Ridgwell et al.* [2003]) tend not to be applicable to shallower water and particularly high organic carbon delivery (especially in the case of CaCO₃ diagenesis) environments, the deep-sea grid is restricted. This can be prescribed 'hard', by defining the sediment grid only at deeper depths and classifying shallowed ocean grid points as invalid (a value of 0.0) in the SEDGEM depth definition file³³ (e.g. *worbe2.depth.36x36x08*). Or, and more flexibly, the entire ocean grid can be defined as potentially valid (e.g. *worbe2.depth.36x36x08.ALL*) and SEDGEM directed to treat any grid points lying shallower than a specific depth as shallow water sediments. This depth cut-off is set via the parameter *sg_par_sed_Dmax_neritic* (in units of m below the ocean surface). A typical value is 176.0³⁴, which for an 8-level configuration will exclude the shallowest ocean depth level from the deep-sea grid, and for a 16-level ocean will exclude the two shallowest ocean depth levels.

Sediments that are not classed as '*deepsea*', are automatically classed as '*mud*', i.e. potentially detrital and organic carbon rich. Such sediment locations currently have no option for carbonate burial associated with them. Instead, a limited range of extremely simple and crude assumptions regarding organic carbon preservation (and P regeneration) are provided.

Locations at which carbonate can be produced and preserved are classified as '*reef*' locations. Note that the assumption here is that carbonate is precipiced benthically and buried, rather than produced pelagically and settle to the seafloor. A mask needs to be provided³⁵ in order to define the cells which are *reef* rather than *mud*. The mask consists of the sediment grid, with values being either 1.0 (*reef*) or 0.0 (other). Grid points that are shallower than the depth cut-off and not associated with a value of 1.0 in the corresponding reef mask file, are classified as *mud*.

If can be a little tedious (not really *that* tedious) to create a *reef* mask than exactly matches all the shallow grid points, so a parameter is provided to force all non *deepsea* grid points to be *reef*. This is achieve by setting:

```
sg_ctrl_sed_neritic_reef_force=.TRUE.
```

This parameter can also be used to force a pattern of *reef* cells as defined by the *reef* mask, even if they lie deeper than the depth cut-off value (*sg_par_sed_Dmax_neritic*).

Additional information and tips on setting up shallow water sediment grids can be found in some of the EXAMPLES (e.g. EXAMPLE.p0251b.PO4.SPINO).

Set a specific ocean chemistry or saturation state

In the absence of a significant pelagic plankton derived deep-sea carbonate sink, the global burial sink of CaCO₃ is likely to have been dominated by shallow water depositional environments. cGENIE can represent something of these systems and the fundamental dynamical difference between deep-sea pelagic and shallow water CaCO₃ sinks (the former is predominantly controlled by preservation and input of C_{org}, whilst the latter is primarily a function of primary production by benthic calcifiers) by specifying particular shallow water grid cells as '*reefal*' (see: 20.15). Also in contrast

³³Set by parameter *sg_par_sed_topo_D_name* that must point to a file in *cgenie.muffin/genie-sedgem/data/input*.

³⁴Note that the simple (esp. CaCO₃ diagenesis schemes) arguably only applicable below depths of ca. 1000 m, although even this is complicated by varying patterns of productivity across the ocean.

³⁵The parameter name to set the reef mask is *sg_par_sed_reef_mask_name*.

to the deep-sea pelagic CaCO_3 sink, which is mechanistically simulated (both export production and particularly early diagenesis and CaCO_3 dissolution within the sediments), the shallow water CaCO_3 sink in cGENIE is heavily parameterized and is treated as little more than a function relating deposition of CaCO_3 (in units of $\text{mol cm}^{-2} \text{ yr}^{-1}$) at each and every designated reefal grid point. A parameter is provided:

```
sg_par_sed_reef_CaCO3precip_sf
```

that scales the CaCO_3 burial flux so as to achieve some desired global value, such as to balance the global weathering rate. Unfortunately, the value of this parameter is not *a priori* known. Its value (and hence sink strength) also depends on the ambient saturation state. Hence, from the outset, the carbonate saturation properties of the ocean surface must be assumed and set. For the modern ocean, this arises naturally from a system initialized with observed concentrations of ALK, DIC, $[\text{Ca}^{2+}]$, etc. and (preindustrial) atmospheric $p\text{CO}_2$, and in conjunction with an adequate simulation of the large-scale productivity of the ocean and recycling in the interior ocean.

For deeper in the geological past and particularly in the absence of an effective deep-sea pelagic CaCO_3 buffer, ALK and DIC in particular *a priori* not known, although either sensitivity experiment assumptions of the CO_2 concentration required to generate a specific climate and/or proxy data, can give some ideas of $p\text{CO}_2$. It is beyond the scope of this HOW-TO to discuss what geological constraints can be applied in constraining surface ocean saturation (and hence, in conjunction with a given weathering flux, to identify the value of the scaling parameter) but having identified a number (as calcite or aragonite), there are three possible ways of setting an appropriate ocean chemistry:

1. To achieve a specific mean ocean surface saturation state: mean ocean ALK and DIC values can be set consistent with assumptions regarding $p\text{CO}_2$ and $[\text{Ca}^{2+}]$. The parameters specifying the initial mean ocean ALK and DIC concentrations ($\mu\text{mol kg}^{-1}$) are:

```
bg_ocn_init_12=2.3630E-03
bg_ocn_init_3=2.244E-03
```

for their respective modern mean ocean values.

In order to estimate appropriate past concentrations for a given $p\text{CO}_2$ and $[\text{Ca}^{2+}]$, trial and error can be employed, aided by use of a carbonate calculator program such as CO2SYS, although bearing in mind that the saturation conditions calculated in CO2SYS will be w.r.t. the surface, while the 2 model initialization parameters are setting the mean ocean composition – ocean circulation and particularly biological productivity in the open ocean will create an offset between mean surface and bulk ocean properties. Once a reasonable ocean saturation state has been employed, the reefal CaCO_3 depositional parameter can be played with to (re)balance weathering and global burial, although if the initial guess is too far off, this will need to be briefly iterative (as localized CaCO_3 removal and burial will affect the local saturation state and hence in turn burial).

Conclusion: do-able, but tedious.

2. An alternative methodology, somewhat akin to how the system with a responsive deep-sea pelagic CaCO_3 buffer is configured, is to set the system 'open', and allow the balance between weathering and shallow water CaCO_3 burial to set ocean chemistry (whilst restoring $p\text{CO}_2$ to a required value). This is potentially even more tedious, because the value of the scaling parameter is not known. In fact, it is the scaling parameter that will set the mean ocean surface saturation in order to balance weathering and shallow water CaCO_3 burial.

An acceleration technique can be used (see: [20.15](#)) but the process is still iterative (and tedious).

A further provision is hence made in the SEDGEM module that will flux force the ocean with Ca (and also DIC if requested) at each and every reefal cell if the local saturation state falls below a specified target value. In this:

`sg_ctrl_sed_neritic_reef_force=.TRUE.`

will turn 'on' the provision of a forcing of ocean chemistry towards a local saturation target,

`sg_par_sed_ohmegamin`

sets the saturation threshold, and

`par_sed_ohmegamin_flux`

specifies the flux in units of mol Ca²⁺ cm⁻² per time-step at each and every reefal grid point.
Setting:

`ctrl_sed_forcedohmega_ca=.false.`

will enable the corresponding fluxes of ALK and DIC to be applied.

This methodology can be accelerated as per Section [20.15](#) but with the downside that what is being set in practice is the minimum saturation at any reefal grid point – i.e. one should obtain a reefal grid point, typically at relatively high latitudes, with ambient chemistry close to the specified saturation, with all other grid points characterized by higher saturation. There is no simple way of deriving the global mean surface saturation in advance.

Having forced ocean chemistry (e.g. just by altering the Ca²⁺ and ALK input to the ocean) and restored to a specified atmospheric pCO₂ value, the value of the reefal CaCO₃ burial scaling parameter can be adjusted in order to balance weathering and global burial. A little iteration may probably be required to get a good balance as the competing flux input and CaCO₃ removal are very localized (at reefal cells).

Conclusion: doable, but does not give a simple-to-predict mean global saturation.

3. With a desired global mean surface saturation value in mind, cGENIE can be configured to determine the appropriate ocean ALK and DIC value directly.

In brief – a mean global saturation target value is set by the parameter:

`bg_par_force_invert_ohmega`

ALK, and if requested: DIC (and isotopes) and Ca, is fluxed evenly throughout the ocean if the current ocean surface saturation value falls beneath the target, and not fluxed at all otherwise³⁶.

An aragonite saturation value can be forced towards rather than calcite by setting:

`bg_ctrl_force_ohmega_calcite=.false.`

The setup for this is a little involved and involves specifying a particular forcing that cGENIE identifies as requiring a saturation target to be matched and requires some additional configuration.

An example *user-config* to spin-up the ocean to a specified saturation value is provided:

³⁶A negative flux of ALK can be enabled, which would act to reduce saturation, by setting the following:
`bg_ctrl_force_invert_nonneg=.false.`

EXAMPLE.p0251b.P04.SPINO

and described in the Examples chapter. A typical configuration would like like the following:

```
bg_par_forcing_name="pyyyzz.RpCO2_Rp13CO2_FRALK_FDIC_F13DIC_FCa"
bg_par_atm_force_scale_val_3=2800.0E-06
bg_par_atm_force_scale_val_4=-6.5
bg_par_ocn_force_scale_val_3=1000.0E12
bg_par_ocn_force_scale_val_4=0.0
bg_par_ocn_force_scale_val_12=2000.0E12
bg_par_ocn_force_scale_val_35=0.0
bg_par_force_invert_ohmega=10.0
```

of which the first line sets the forcing (provided), the second line specifies the atmospheric $p\text{CO}_2$ value to be restored to (and the 3rd its isotopic composition). Because the ocean is either fluxed or not, depending on surface saturation compared to the target, how aggressively the ocean is fluxed is set by lines #4 (for DIC) and #6 for ALK. The values shown here³⁷ are approximately $\times 10$ global weathering for reference. The last line is the required mean ocean surface saturation state. (Lines #5 and #7 set the isotopic composition of injected DIC, and any associated Ca flux, respectively.)

In terms of methodology:

- (a) This saturation restoring would be carried out in a closed system for typically 10 or 20 kyr, depending on whether an initial *spin-up* was being used and how far off any guess as to initial ALK and DIC is.
Depending on the reported global CaCO_3 burial rate³⁸, the reefal CaCO_3 burial scaling parameter is adjusted. This is also a good time if climate-dependent feedback is to be used, to adjust e.g. the baseline mean global temperature.
- (b) The saturation restoring *forcing* is replaced by a simple atmospheric $p\text{CO}_2$ *forcing* (or none at all), and a short, perhaps just 1 kyr experiment is carried out with an open system. Fine-tuning of the CaCO_3 burial scaling parameter is carried out (plus any fine-tuning required to set an exact initial weathering flux).
- (c) A long, open-system *spin-up*, likely accelerated (Section 20.15) 3rd spin-up phase is conducted with no forcing.

Example *user-configs* of this sequence are provided (EXAMPLE.p0251b.P04.SPIN*). Note that the 3rd phase of spin-up could be carried out with a prescribed $p\text{CO}_2$ *plus* associated isotopic composition if not already done so (e.g. as part of the phase #1 of the spin-up).

Conclusion: work-able.

Specify a particular carbonate mineralogy

Firstly – pelagic carbonate production and deep-sea sedimentary diagenesis is inherently assumed to be (all) calcite. Currently there is no alternative option, nor mixed phase (including high-Mg calcite) assemblage option.

Shallow water (neritic) carbonates, however, can be specified as either calcite (the default) or aragonite, the difference being simply in which saturation state is used to calculate precipitation and burial rate. Reefal carbonate precipitation assumed in the form of aragonite is simply set by:

```
sg_par_sed_reef_calcite=.false.
```

³⁷These values represent the *total* flux that distribution throughout the ocean relative to grid cell volume.

³⁸SEDGEM file seddiag_misc_DATA_GLOBAL.res

Obviously, the same scaling value as per for calcite cannot be used (aragonite will require a higher scaling to achieve the same global CaCO_3 depositional flux).

In terms of a spin-up to a specified saturation state (see: [20.15](#)) – an aragonite, rather than calcite (the default) saturation value can be forced towards rather than calcite by setting:

```
bg_ctrl_force_ohmega_calcite=.false.
```

Set up a (silicate) weathering feedback

To create a temperature (only) dependency for the weathering of carbonate and silicate rocks, the following two parameter values need to be set (for carbonate and silicate weathering, respectively):

```
rg_opt_weather_T_Ca=.true.  
rg_opt_weather_T_Si=.true.
```

A reference temperature governs the dependency of weathering on climate change and modifies the solute fluxes from weathering scaled (non-linearity) to the deviation of mean global land surface temperature from the reference temperature. The mean global land surface temperature is given in the **BIOGEM time-series** file `biogem_series_misc_SLT.res` and is set equal to the reference temperature parameter:

```
rg_par_ref_T0.
```

The baseline (unmodified) solutes fluxes from terrestrial weathering are set by the parameters:

```
rg_par_weather_CaCO3  
rg_par_weather_CaSiO3
```

for carbonate and silicate weathering, respectively. At steady-state (no climate perturbation), these must sum up to the total weathering flux which is given by the total global CaCO_3 burial flux found in the file `seddiag_misc_DATA_GLOBAL.res` (in the `genie-sedgem` results sub-directory). To balance the silicate weathering, volcanic CO_2 out-gassing is then assigned a value equal to silicate weathering flux by setting the parameter `rg_par_outgas_CO2`.

There are different ways in which the total weathering flux (equal to total CaCO_3 burial at steady state) can be split between carbonate and silicate weathering and hence a value for volcanic CO_2 out-gassing assigned:

1. All solutes could simply be assumed to be derived from carbonate weathering (which is the default assumption in open system experiments without a weathering feedback), e.g.:

```
rg_par_weather_CaCO3=10.0E+12  
rg_par_weather_CaSiO3=0.0
```

for a hypothetical example with 10 Tmol yr^{-1} total global CaCO_3 burial. No volcanic CO_2 out-gassing should then be prescribed.

2. Secondly, silicate and carbonate weathering could be assumed to be split evenly, e.g.:

```
rg_par_weather_CaCO3=5.0E+12  
rg_par_weather_CaSiO3=5.0E+12
```

Volcanic CO_2 out-gassing needs to be set equal to the baseline silicate weathering flux:

```
rg_par_outgas_CO2=5.0E+12
```

3. Thirdly, volcanic CO_2 out-gassing could be assumed, which then constrains the silicate flux, with the carbonate flux being whatever is required to make the total Ca^{2+} weathering flux equal to global CaCO_3 burial.

A difficulty arises because setting the reference temperature `rg_par_ref_T0` equal to the mean global land surface temperature only gives rise to weathering fluxes exactly equal to the reference parameter values (`rg_par_weather_CaCO3` and `rg_par_weather_CaSiO3`) for a non-seasonally forced climate. Because weathering is non-linear in climate (here just temperature), a seasonally forced configuration of the model will give rise to slightly modified weathering fluxes. The result is a slight drift in atmospheric $p\text{CO}_2$ and climate, even after a fairly long (100s of kyr), or alternatively, a slightly different steady state $p\text{CO}_2$ value (for a ca. 1 Myr *spin-up*).

The reason is that long-term climate and hence $p\text{CO}_2$ is controlled by the silicate weathering component of total weathering, not total weathering. Under a seasonally forced rather than annual average climate, the non-linearity in the weathering response to temperature deviations means that silicate weathering will generally slightly exceed volcanic CO_2 out-gassing. Atmospheric $p\text{CO}_2$ and with it global temperatures will hence be gradually drawn-down until net (carbonate) carbon removal exactly matches the rate of new (volcanic) carbon input.

The imbalance between silicate weathering and volcanic CO_2 out-gassing is given in the BIOGEM output:

`biogem_series_misc_exweather_Ca.res`

which records the absolute excess of weathering compared to out-gassing, in mol Ca^{2+} yr^{-1} as well as a percentage. In an *open system* (not a closed one!), this excess needs to be adjusted 'close' (how close? sub 1 percent or 0.1 Tmol Ca^{2+} yr^{-1} , certainly) to zero. This adjustment is done by running a short (a few or 10s of years) experiment, reading off the excess weathering value, and doing one of the following:

1. Adjust the reference temperature parameter `rg_par_ref_T0` – to a lower value if there is an excess of silicate weathering over out-gassing, or
2. Adjust the reference silicate weathering parameter `rg_par_weather_CaSiO3`, or
3. Adjust the volcanic CO_2 out-gassing flux parameter `rg_par_outgas_CO2`.

An exact adjustment can be made for the second and third possibilities, with the easiest being to reduce the out-gassing flux value by the amount of excess weathering. For the first option, a couple of iterations will typically be required in order to determine the new reference temperature value that gives rise to a silicate weathering balance. The second option is the recommended one, with the third being the least recommended. Regardless of which of the 3 options, the total weathering flux, given in the BIOGEM time-series file `biogem_series_diag_weather_Ca.res`, will now be slightly different from the required burial flux. This either requires that the value of the reference carbonate weathering parameter (`rg_par_weather_CaCO3`) is now slightly adjusted, or that a slightly different global carbonate burial flux is allowed.

If the carbon isotopic signature of volcanic CO_2 out-gassing is assigned a value of e.g. -6.0‰. The $\delta^{13}\text{C}$ of weathered CaCO_3 is then simply set in order that inputs equal the mean $\delta^{13}\text{C}$ of carbonate burial, which as given in the **BIOGEM time-series** file:

`biogem_series_sed_CaCO3_d13C.res` For example, assuming equal fluxes of 5 Tmol yr^{-1} for both weathering components and for volcanic CO_2 out-gassing (at -6.0‰) and assuming a mean carbonate burial $\delta^{13}\text{C}$ of 3‰, requires:

```
rg_par_outgas_CO2_d13C=-6.0
rg_par_weather_CaCO3_d13C=12.0
```

Of course, in reality organic carbon burial is important and including it would enable a much more realistic value of weathered carbonate $\delta^{13}\text{C}$ to be set ...

Accelerate the weathering-sedimentation mass balance ('GEMlite')

Also see: The FAQ Chapter.

A (pseudo) module is provided: 'GEMlite' which provides a means of much more rapidly solving the weathering-sedimentation mass balance – i.e. the long-term (>10 kyr) carbon cycle processes and feedbacks. The motivation behind GEMlite is the stark disparity between the time-scales of ocean circulation and biological pump (ca. 0.1-1000 years) and those of sedimentation and weathering (2-20 kyr) and particularly the silicate weathering feedback (>100 kyr). This makes running cGENIE to an open system steady state (with or without the silicate weathering feedback) challenging. Is there any way of 'accelerating' the calculation of the 'long tail' [Archer et al., 2009] of the CO₂ curve (e.g. in response to fossil fuel CO₂ emissions)?

The philosophy is as follows: the long-term weathering-sedimentation processes are effectively just an imbalance between the supply of solutes via weathering and preservation and burial of esp. carbonates in deep-sea (and shallow) marine sediments. For a small imbalance between weathering and sedimentation, atmospheric pCO₂ and climate (and hence the solute flux when including weathering feedbacks) will only change very slightly. For long intervals characterized by only a small imbalance in weathering-sedimentation the key assumption is made: Ocean circulation and the biological pump, and hence the *gradients* of dissolved species in the ocean can be considered *invariant*. Hence, for the purpose of solving weathering-sedimentation over an intervals of time: The ocean can be treated as a *single box*. It further assumes that: The ocean is initially in equilibrium with the atmosphere (w.r.t. CO₂). (This latter assumption does place important limitations on under what circumstances GEMlite can be employed to accelerate experiments.)

This is what GEMlite does – it solves for weathering-sedimentation and applies the mass difference *uniformly* throughout the ocean (as if it were a single box), hence preserving the tracer gradients in the ocean. It also (optionally) calculates and re-partitioning of carbon between ocean and atmosphere. Because ocean circulation and the biological pump etc. do not have to be recalculated, the accelerated quasi box-model phase can be calculated very considerably faster than the 'full' model. Obviously, if atmospheric pCO₂ and hence climate are changing at an appreciable rate then the assumption of invariance in ocean tracer gradients breaks down and it is not 'safe' to apply the accelerated calculation. Similarly, appreciable changes in nutrient inventories will affect the biological pump and hence also change tracer gradients.

The key to employing GEMlite, in addition to knowing when it is appropriate/not appropriate to employ it, is to decide what balance of accelerated (GEMlite) time-stepping vs. normal (full system update of ocean circulation, biological pump, etc.) time-stepping to employ. This division is implemented by creating a sequence of accelerated vs. non-accelerated time-stepping. This can be done in one of two ways:

1. Fixed sequence.

By default, GEMlite will employ a fixed, pre-determined sequence of accelerated vs. non-accelerated time-stepping. The parameters to specify this sequencing are:

`ma_gem_notyr` – which sets the number of years (the assumed time-step of GEMlite) for 'normal' time-stepping.

`ma_gem_yr` – which sets the number of years for accelerated time-stepping.

For instance: if `ma_gem_notyr=50` and `ma_gem_yr=50`, you would have a sequence with 50 years of full updating, followed by 50 years of accelerated.

For instance: if `ma_gem_notyr=10` and `ma_gem_yr=90`, you would have a sequence with 10 years of full updating, followed by 90 years of accelerated.

etc.

Note that the GEMlite cycle phase of 'normal' time-stepping is *always* done first.

Also note that choosing e.g. `ma_gem_notyr=10` and `ma_gem_yr=100`, while appearing a desirably simple ratio, would result in the change-over point in cycle phase (to accelerated) occurring at the end of year 10, 120, 230, 240, etc. – something that might affect/influence your choice of data saving pattern (i.e., the sequence of time-points for time-series and time-slice data saving).

By default, the parameter values are: `ma_gem_notyr=999999` and `ma_gem_yr=1` meaning that in practice you will never get to the end of the 'normal' time-stepping phase. Note that these parameters are **integers** (setting real numbers, e.g. `1.0E6` will not work ...).

2. Adaptive sequencing.

Here, GEMlite attempts to be clever and optimizes the ratio between the duration of each phase of the cycle.

The motivation for this is that often in model experiments, environmental parameters will tend to change faster at the beginning of an experiment compared to towards the end. Fossil fuel CO₂ release and its long tail of declining pCO₂ is a good example of this. Obviously this complicates the choice of a (fixed) ratio of cycle phases – 100:100 (or more likely: 1000:1000) might not lead to too much degradation of the simulation, but you would only gain a speed advantage of x2 for the experiment as a whole, which if 100-1000 kyr in total duration, is still going to be long. On the other hand: 10:90 would give you a factor almost x10 increase in overall speed, but would seriously degrade the simulation during the initial, rapidly changing environment following CO₂ release.

Adaptive sequencing adjust the time-stepping via 2 criteria:

- In the normal time-stepping phase, if the rate of change of pCO₂ is *more than* a specified threshold over any one year, then the total duration of this phase is extended by one year.
- In the accelerated time-stepping phase, if the total change in pCO₂ since the last normal phase is *less than* a specified threshold, then the total duration of this phase is extended by one year.

The result is that the phase durations are always a minimum of the values set by `ma_gem_notyr` and `ma_gem_yr`. If it is 'unsafe' to switch to accelerated mode, because pCO₂ is changing rapidly, then the model stays in normal mode. If it is safe to stay in the accelerated mode, because pCO₂ has not changed much in total during the phase, then the model stays in the accelerated phase.

The parameter names are default values for the two thresholds are:

- `ma_gem_adapt_dpCO2dt=0.1` (ppm yr-1)
- `ma_gem_adapt_DpCO2=1.0` (ppm)

but these will not necessarily be the ideal of any particular experiment (and some trial-and-error may be called for).

Adaptive time-stepping is enabled by setting:

`ma_gem_adapt_auto=.true.`

(by default it is `.false.`).

The switching between normal (non accelerated) and accelerated phases is saved in a time-series file:

```
biogem_series_misc_gemlite.res
```

As a further refinement, the accelerated phase can be set to be relatively short to begin with, but gradually increasing in length. The parameters controlling this are:

`ma_gem_yr` – the initial accelerated phase duration

`ma_gem_yr_max` – the maximum accelerated phase duration

`ma_gem_adapt_dgemyr` – the (minimum) fractional increase in duration each cycle (or 1.0 yr, whichever is greater)

A reasonable set of parameters:

```
ma_gem_notyrs=10
ma_gem_yr=10
ma_gem_yr_max=990
ma_gem_adapt_dgemyr=0.05
ma_gem_adapt_dpCO2dt=0.10
ma_gem_adapt_DpCO2=0.01
ma_gem_adapt_auto=.true.
ma_gem_adapt_auto_unlimitedGEM=.false.
```

Finally ... you will need a *base-config* that has GEMlite enabled. This actually requires nothing more than the addition of a couple of lines (to a *base-config* file):

```
ma_flag_gemlite=.TRUE.
```

which can go e.g. near the start of the file under # GENIE COMPONENT SELECTION. Plus:

```
ma_kgemlite=xx
```

which can go e.g. under # TIME CONTROL AND TIME-STEPPING.

Here, xx will depend on the time-step assumed in the base-config. This is likely to be either 96: the standard for most *base-configs*, or 48: for low resolution and faster model configurations, which typically have .t48 in their filename. By convention, I name *base-configs* including GEMlite with _gl,

e.g. cgenie_eb_go_gs_ac_bg_sg_rg_gl.p0000c.BASESLi.t48.config
but you can name it BobTheLeglessPony for all I care.

The **most important** thing is to ensure you are not seriously degrading model fidelity (of carbon cycle simulation) by your adoption and configuration of GEMlite.

Test different assumptions of how the time-stepping phases are scheduled and compare (of possible) against a full experiment in which GEMlite is not used.

It is important to recognize that when the model switches into the GEM phase, it assumes all ocean tracer gradients are fixed, and updates only ocean composition as a whole according to weathering vs sedimentation imbalance (and also tries to re-equilibrium ocean and atm). As part of this, the flux to the sediments is taken from the average of the last year of the preceding normal phase, and fixed. This also means that the d13C of the CaCO₃ deposited to the sediments is fixed ... even if the ocean d13C is being updated and changing ... So, basically you lose the feedback that leads to d13C converging as sinks balance (weathering and volcanic) inputs³⁹.

The solution is to not run in the GEM phase for such long intervals – instead giving the normal

³⁹Adjusting the fluxes themselves during the GEM intervals would break the underlying assumption inherent in the acceleration approximation.

phase a chance to make a brief update of ocean gradients and also d13C of export flux. BUT, if pCO₂ hardly changes, cGENIE runs the risk of staying in the GEM phase for ever (ish)!

A further option:

`gem_adapt_auto_unlimitedGEM`

sets whether GEM is allowed an unlimited phase duration or not. By default it is `.false.`. This means that the maximum GEM duration is limited to the normal `gem_yr` parameter. Also, if excessive (pCO₂) drift occurs, the model will immediately switch to the normal phase.

By default then:

- `gem_notyr` specifies a MINIMUM duration for a normal phase..
- `gem_yr` specifies a MAXIMUM duration for a GEM phase.

Values of `gem_yr` much less than 100 are not advisable as you will not reestablish a new equilibrium gradient of tracers in the ocean in that time.

20.16 HOW-TO ... geoengineering

Set up carbon dioxide removal (CDR) geoengineering experiments

There are various ways implement different carbon dioxide removal strategies in cGENIE. For instance – additions of dissolved iron and phosphate can be implemented as simple flux forcings to the ocean surface, as described in the Tutorial exercises. Similarly, ocean ‘liming’ can be implemented as a flux *forcing* of alkalinity to the surface (with or without associated Ca²⁺ and with or without additional CO₂ emissions to the atmosphere due to the creation of lime). There is also the facility for automatically calculating the liming required for a specific policy target (atmospheric CO₂ history or desired mean ocean surface pH or saturation state). This is described in a subsequent HOWTO. This section describes a framework created for applying additional geoengineering modifications and particularly ones that cannot be implemented as a simple flux *forcing*.

Applying a geoengineering ‘liming’ flux to the ocean surface

There are two different methodologies provided for: (1) applying a surface flux of alkalinity (and Ca, and DIC) with a uniform or spatial pattern, regardless of atmospheric *pCO₂* and emissions, and (2) applying a surface flux calculated internally to meet some objective – here, the value of atmospheric *pCO₂* at any point in time.

1. The *forcing*:

`pyyyyz_RpCO2_Rp13CO2_FRALK_FDIC_F13DIC_FCa`

provides a facility for applying alkalinity (ALK) (as well as Ca, and DIC, if selected) to the ocean surface concurrently with a prescribed time-history of restoring of atmospheric *pCO₂* (and associated $\delta^{13}\text{C}$), while

`pyyyyz_FpCO2_Fp13CO2_FRALK_FDIC_F13DIC_FCa`

is similar, except with a prescribed time-history of flux (emissions) of CO₂ (and associated isotopic composition) to the atmosphere. By default in each, only an ALK flux to the ocean surface is selected. The other three ocean tracers listed in `configure_forcings_ocn.dat` are set to ‘f’ for flux forcing. As provided in these examples, atmospheric pCO₂ is held at 278 ppm for the restoring, and emissions of 1 PgC yr⁻¹ (8.333e+013 mol yr⁻¹) for the flux forcing version.

By default, *forcings* are set such that the ALK (and Ca, and DIC if selected) is applied uniformly over the ocean surface (the ‘2’ in COLUMN #06). Point sources can be specified by changing this to a ‘0’, or an explicit spatial pattern with a ‘-2’ (which must then be provided in a separate file).

2. The *forcing*:

`pyyyz_FRpCO2_Fp13CO2_FRALK_FDIC_F13DIC_FCa`

differs firstly in specifying both a CO₂ emissions flux and restoring value for atmospheric pCO₂. For alkalinity (ALK), both a flux and restoring of the ocean are selected.

Basically, what happens here is when a flux + restoring of ocean ALK is selected together with flux + restoring of pCO₂, ALK additions to the ocean is made in order to try and maintain the prescribed history of atmospheric pCO₂, regardless of the CO₂ emissions also specified. If at any one time-step atmospheric pCO₂ is higher than the target value, ALK is added to the ocean with the flux specified in the ALK flux forcing. If atmospheric pCO₂ is lower than the target value, no ALK is added or taken away, unless the parameter:

`bg_ctrl_force_invert_noneg = .true.`

is set, which enables a negative ALK flux to be applied⁴⁰.

⁴⁰This is likely pretty un-physical for most applications, hence the default is `.false.`.

Note that the restoring value of ALK has no meaning and the value set in:

```
biogem_force_restore_ocn_ALK_sig.dat
```

is not important. Both atmospheric CO_2 flux forcing and restoring forcing time-series are specified 'as normal', and may constitute an SRES CO_2 emissions scenario and RCP based pCO_2 time-history, respectively, for example. For d13C, only the d13C of emissions is specified⁴¹.

If an atmospheric CO_2 emissions is not required, simply set the value in the time-series file to zero. Note that no scaling of the atmospheric CO_2 forcing⁴² must be applied because it will scale both restoring and emissions ...

If the model cannot quite attain the pCO_2 target specified, you probably do not have a sufficiently large ALK flux specified. Conversely, if the pCO_2 target is significant over-shot (technically: under-shot) then you might have prescribed too large a flux.

Geoengineering with ... 'pipes'

Pipes are parameterized following *Yool et al. [2009]* (Yool, A., J. G. Shepherd, H. L. Bryden, and A. Oschlies (2009), Low efficiency of nutrient translocation for enhancing oceanic uptake of carbon dioxide, *J. Geophys. Res.*, 114, C08009, doi:10.1029/2008JC004792.) This is selected by setting:

```
bg_opt_misc_geoeng='pipes'
```

(Currently, there is no other option and anything passed other than a value of 'pipes' results in the default: no geoengineering.) A series of option then control the working of the pipes:

- A mask file is provided to designate the grid points of the ocean with pipes in them, via:

```
bg_par_misc_2D_file
```

with a default of 'misc.dat'. The default file location is:

```
cgenie.muffin/genie-biogem/data/input.
```

This file is treated in a similar way to the normal 2D *forcing* files. The values at each grid point can be scaled via the parameter: `bg_par_misc_2D_scale`. The units are m³ per year. e.g. setting `bg_par_misc_2D_scale=1E13`, assuming values of 1.0 or 0.0 in `misc.dat` will create an annual vertical advective flux at each grid point equivalent to 30% of the volume of the surface cell (3.2E13).

- The ocean depth level associated with the base of the pipes is set via:

```
bg_par_misc_kmin_pipe=12
```

- Three parameters are then provided to control what is advected, with a number of combinations of tracers possible (useful for diagnosing the relative importance of e.g. nutrients vs. respired CO₂ vs. temperature and salinity (and hence ocean circulation changes)):

– # pump T and S?

```
bg_ctrl_force_GOLDSTEInTS=.false.
```

(the default) prevents T and S being advected.

⁴¹A restoring must *not* also be set.

⁴²e.g. `bg_par_atm_force_scale_val_3`

```

- # ONLY pump T and S?
  bg_ctrl_force_GOLDSTEInTSonly=.true.

  results in *only* T and S being advected. This requires that bg_ctrl_force_GOLDSTEInTS=.false..
  Its default is .false..
- # pump no DIC?
  bg_ctrl_misc_geoeng_noDIC=.false.
  prevent DIC from being advected.

```

The following combinations are then valid (shown commented out (#) are the settings that are the same as the default settings and hence do not need to be re-defined, although it would not hurt to):

1. bg_ctrl_force_GOLDSTEInTS=.true.
~~#bg_ctrl_force_GOLDSTEInTSonly=.false.~~
~~#bg_ctrl_misc_geoeng_noDIC=.false.~~
 results in everything being advected.
2. ~~#bg_ctrl_force_GOLDSTEInTS=.false.~~
~~#bg_ctrl_force_GOLDSTEInTSonly=.false.~~
~~#bg_ctrl_misc_geoeng_noDIC=.false.~~
 results in everything (nutrients, DIC, ALK, isotopes, etc.) except T and S being advected.
3. ~~#bg_ctrl_force_GOLDSTEInTS=.false.~~
~~#bg_ctrl_force_GOLDSTEInTSonly=.false.~~
 bg_ctrl_misc_geoeng_noDIC=.true.
 results in everything except T and S *and* DIC being advected (i.e. just nutrients, alkalinity, isotopes etc.).
4. bg_ctrl_force_GOLDSTEInTS=.true.
 bg_ctrl_force_GOLDSTEInTSonly=.true.
~~#bg_ctrl_misc_geoeng_noDIC=.false.~~
 results in only T and S being advected.

Combination #1 is arguably the only realistic setting – the others being for diagnosing how the model works and the primary controls on the effectiveness or otherwise of pipes, only. The difference between #2 and #1 indicate the importance of changes in ocean circulation driven by T and S, which can also be assessed in isolation via option #4. The difference between option #3 and #2 indicates the importance of the respired CO₂ 'leak' in the effectiveness of ocean pipes. (Note that there is no option for removing DIC only and e.g. advecting T and S and nutrients etc etc.)

Adjust solar forcing in a time-dependent manner

The value of the solar constant in cGENIE is set by the *namelist parameter*:

ma_genie_solar_constant

which by default is set to 1368 W m⁻², i.e.:

ma_genie_solar_constant=1368.0

Specifying a different value for `ma_genie_solar_constant` in the *user-config* file allows the solar forcing of the EMBM to be altered. For example, to induce a 'snowball Earth' like state under a solar constant applicable to the late Neoproterozoic (some 6% less than modern) you would set:

```
ma_genie_solar_constant=1330.56
```

Modification of `ma_genie_solar_constant` can be turned into a time-dependent forcing of solar forcing, but only by frequent re-starting using a sequence of short model integrations.

Alternatively, a crude (temporary) hack is provided to allow a semi-continual adjustment of solar forcing. Whether you wish to vary the solar constant in a time-dependent manner is determined by the *parameter*:

```
bg_ctrl_force_solconst
```

By default this is set to `.false.`. By adding to the *user-config* file:

```
bg_ctrl_force_solconst=.true.
```

a time-varying change in the value of the solar constant will be imposed. For this, **BIOGEM** will expect the presence of a file called `biogem_force_solconst_sig.dat` in the forcing directory⁴³. This file must contain two columns of information: the first is a time marker (year) and the second is a paired value for the solar constant. In the current crude incarnation of this feature, the time markers (1st column) **must** correspond exactly to the time markers in the time-series specification file⁴⁴. **cGENIE.muffin** will exit with an appropriate error message if this is not the case.

When using the time-varying solar constant hack, seasonal solar insolation is re-calculated each year with a call to `radfor(genie_solar_constant)`⁴⁵ at the start of the time-stepping loop⁴⁶. At each time-marker, **BIOGEM** sets the value of `genie_solar_constant` equal to the corresponding value specified in `biogem_force_solconst_sig.dat`. Thus, regardless of how closely-spaced the time-marker years are, (seasonal) solar insolation is only adjusted every year. For a longer time-marker interval than yearly, no interpolation is performed on the series of solar constant values, and in this way time-dependant solar forcing currently differs from the calculation of other *forcings*.

A simple example file might look something like:

```
-START-OF-DATA-
0.5 1367.0
1.5 1366.0
2.5 1365.0
3.5 1364.0
4.5 1363.0
5.5 1362.0
6.5 1361.0
7.5 1360.0
8.5 1359.0
9.5 1358.0
10.5 1357.0
-END-OF-DATA-
```

⁴³REMEMBER: The location of which is specified by the namelist parameter `bg_par_fordir_name`.

⁴⁴REMEMBER: The filename of which is specified by the namelist parameter `bg_par_infile_sig_name`.

⁴⁵`cgenie.muffin/genie-embm/src/fortran/radfor.F`

⁴⁶`cgenie.muffin/genie-main/genie.F`

which will decrease the value of the solar constant by 1 Wm^{-2} each year. Note that because the solar forcing is only updated each year (with the call to `radfor.F`), the first year will be characterized by climate with a solar constant of 1368 Wm^{-2} , the default. Although **BIOGEM** sets a new value of `genie_solar_constant` (1367 Wm^{-2}) mid way through the first year, it is only at the start of the second year that solar insolation is recalculated according to the reduction in solar constant.

Hacking the solar constant in a time-varying manner is, of course, a (albeit crude) way of addressing SRM geoengineering impacts.

20.17 HOW-TO ... orbits

Specifying a fixed orbital configuration

The parameters to prescribe a fixed (but different from modern) orbital configuration are:

- ea_par_orbit_osce=0.0167
- ea_par_orbit_oscob=0.397789
- ea_par_orbit_oscgam=102.92

which specify, respectively:

- eccentricity
- sine of obliquity
- longitude of perihelion

and here are shown with their modern, default values.

To specify an alternative (different from modern) orbital configuration, such as for the last glacial maximum (21 ka)⁴⁷, you would add (to the *user-config*):

- ea_par_orbit_osce=0.018994
- ea_par_orbit_oscob=0.389911
- ea_par_orbit_oscgam=114.42

For comparison, the mid-Holocene (6 ka) parameter set⁴⁸ is:

- ea_par_orbit_osce=0.018682
- ea_par_orbit_oscob=0.408410
- ea_par_orbit_oscgam=0.87

Specifying time-varying orbits

For model simulations over long timescales you might want to consider the effects of changes with time in the astronomical forcing. In this case, the amount of daily mean insolation at each latitude can vary as a function of eccentricity, obliquity and (climatic) precession, as opposed to under a fixed insolation pattern. Such transient forcing can be applied in **muffin** by adding the following set of parameter settings to the *user-config* file:

```
# Call orbit_radfor: Applies astronomical forcing
ea_38 = "y"
# Specify the type of orbital forcing: 0 (default), 1 (time-varying)
ea_39 = 1
# Number of data points in orbits file
ea_40 = 1001
# Interval between data points in units of goldstein time steps
ea_41 = 96000
# filename for orbital parameters
ea_42 = "orbits_La2004_1Myr.dat"
```

In this specific example, a file: *orbits_La2004_1Myr.dat* is specified, which provides the astronomical solution for the past 1 million year based on the astronomical solution of *Laskar* (2004) in time-steps of 1,000 years.⁴⁹ For 1 million year in time-steps of 1,000 years, there are 1,000,000 / 1,000 = 1,000 data points (+ 1 for year zero) = 1,001 data points total. The value of parameter

⁴⁷Taken from: [PMIP2](#) experiment boundary conditions.

⁴⁸Taken from: [PMIP2](#) experiment boundary conditions.

⁴⁹The data file .dat can be found in sub-directory: *genie-embm/data/input*

`ea_40` needs to be set equal to this value, i.e. prescribing how many data points are present in the file specified by `ea_42`.

Note that the file containing the astronomical solution is read in reverse order, i.e. from the bottom to the top. Thus, for file `orbits_La2004_1Myr.dat`, the values for eccentricity and obliquity used for the first 1000 year of the simulation, are 0.03576 and 0.40080, respectively, corresponding to the astronomical solution 1 million year ago. To change the time slice that you would like to simulate⁵⁰ (for example from 500 ka to recent), simply copy the astronomical solution into a new text file and remove the last 500 lines so that the simulation begins with the solution at 500 ka.

The value for parameter `ea_41` depends on the resolution of the model that is used. By default, for a 16-level ocean circulation model configuration, **muffin** employs 96 time-steps per year in the **GOLDSTEIN** ocean circulation model component. When the astronomical solution is provided as one orbital value per 1000 years, this equates to $1000 \times 96 = 96000$ **GOLDSTEIN** time steps. For lower resolution configurations of **muffin**, **GOLDSTEIN** may be operating on 48 time steps per year, which means that `ea_41` need to be changed to 48000 (when the astronomical solution is provided 1000 year time steps).

Important. When using one of the paleo configurations of **muffin** defined in the `genie-paleo` directory, the file containing the astronomical solution must be present in that directory. For all other configurations, the file lives in `genie-embm/data/input`. A warning message will be generated and the **muffin** will exit if the file is not found in the correct directory.

Saving orbitally-relevant data

The challenge here is that changes in seasonal characteristics are critical to understanding orbital modulated insolation forcing and response, so seasonal or even monthly resolved data is needed ... yet the experiments may be several 100 kyr to Myr in duration ... Saving seasonal or monthly resolved netCDF sufficiently frequently to resolve precession – every few kyr – is possible, but the resulting data file size maybe near unmanageable, and in any case, the net CDF files would have to be laboriously processed. The situation can be helped a little by only saving 2D netCDF (see earlier), but it still remains a task to process a large seasonal/monthly netCDF file. In contrast, time-series data is cheap to save, but is currently saved only as global and annual average, masking the details of any orbital response.

The solution provided here is to save data at specific points in the model grid (that are specified prior to an experiment being run), and to be very specific about what data is saved (i.e. rather than large generic sets of variables being saved as in the time-series and time-slice data saving approaches, only certain variables are saved). Furthermore, because orbital variability is slow compared to the seasonal cycle, the data can be aliased. i.e., rather than saving every time-step during the seasonal cycle, and saving every year (of every seasonal cycle), one can devise an interval of data save, that samples a complete seasonal cycle only when x (where $x > 1$) complete years have been simulated. For example – saving data every 13 months would mean that Jan is saved in year 1, Feb in year 2, Mar in year 3, etc, with all months saved over the course of 12 years and then Jan again in year 13. This would not work so well if significant inter-annual variability was present (i.e. as per in a fully coupled model), but in the basic EMBM-based configuration of **muffin**, there is no inter-annual variability and hence this strategy should work just fine.

⁵⁰Don't forget to update parameter `ea_40`!

To configure this way of saving data requires that two files are present in the: genie-biogem/data/input sub-⁵¹ directory, are identified by name via 2 new *user-config* parameters:

1. `bg_par_infile_orb_pts_loc_name`
which directs **BIOGEM** to a file containing a list of model grid locations, in (i,j) format, and
2. `bg_par_infile_orb_pts_var_name`
which directs **BIOGEM** to a file containing a list of variables to be saved.

The default setting of `bg_par_infile_orb_pts_loc_name` is file: `save.orb_pts_loc.EXAMPLE.dat`, whose content looks like:

```
01 01
01 02
01 35
01 36
```

For `bg_par_infile_orb_pts_var_name` is file: `save.orb_pts_var.EXAMPLE.dat`, and whose content looks like:

```
ipoa_solfor.
ipoa_fxsw.
ipoa_seaice.
ipoa_cost.
ocn_sur_temp.
ocn_ben_temp.
```

BIOGEM will save *each* variable at *each* location, saving the data in files with names⁵²:

```
biogem_orb_i01j01.res
biogem_orb_i01j02.res
biogem_orb_i01j35.res
biogem_orb_i01j36.res
```

The data is written out on each time-slice save step. Internally, in between time-slice saves, **BIOGEM** can store up to a maximum number of data points, determined by the variable:

```
bg_n_orb_pts_nmax=0
```

By default, this is set to zero, and **BIOGEM** will therefore not save orbital data, nor look to see if the files `bg_par_infile_orb_pts_loc` and `bg_par_infile_orb_pts_var_name` even exist.⁵³ **BIOGEM** will warn you if you run out of internal storage, should you have e.g. too low a value of `bg_n_orb_pts_nmax` and/or too infrequent a *time-slice* save.⁵⁴

The frequency of data sampling is given in units of **GOLDSTEIN** time-steps, and is set by the parameter:

```
ma_conv_kocn_korb=999999999
```

(the default value, effectively disabling saving).

⁵¹Regardless of whether or not a paleo configuration is used.

⁵²(for this example)

⁵³This means that for non orbital data saving, you do not even need to have these default files present.

⁵⁴I would not recommend exceeding a product of `bg_n_orb_pts_nmax` X number of locations X number of variables of ... 10,000,000 (whether disk space exists to save a long run, is another matter)

For example⁵⁵:

- `ma_conv_kocn_korb = 2`
would sample each and every **BIOGEM** time-step (48 per year) for the standard 16 level ocean (96 time-steps per year in the ocean model).
- `ma_conv_kocn_korb = 96`
would sample once per year for the standard 16 level ocean (96 time-steps per year in the ocean model).
- `ma_conv_kocn_korb = 98`
would alias the sampling, so that each year, you sample one **BIOGEM** *time-step* later in the year. So after 48 years, you will have sampled each of the **BIOGEM** *time-steps* once (and hence have sampled the variability in a year), but only have 48 lines of data, rather than 48*48 lines for 48 years sampling each and every **BIOGEM** time-step.

For the variable list – the format is similar to the netCDF data names. If the sub-string 'sur' appears in the variable string, surface data is saved. If the sub-string 'ben' appears in the variable string, then benthic data is saved.

You can pick any of the ocean tracers listed in `tracer_define.ocn` as part of the string (assuming they are even selected and simulated ...)⁵⁶, e.g.

```
ocn_sur_P04.  
ocn_sur_DIC.  
ocn_ben_sal.
```

Carbonate variables can also be chosen – the full list of possibilities is:

```
H.  
fug_CO2.  
conc_CO2.  
conc_CO3.  
conc_HCO3.  
ohm_cal.  
ohm_arg.  
dCO3_cal.  
dCO3_arg.  
RF0.
```

2D atm-ocn physics variables (no sur or ben) include:

```
seoice.  
seoice_th.  
solfor.  
fxws.  
cost.  
KC02.  
MLD.  
evap.  
precip.
```

⁵⁵Likely you'll want a larger value of `ma_conv_kocn_korb` than the above examples, because it is still effectively saving some data each and every year.

⁵⁶Note that if you select an isotope tracer, e.g. the ¹³C tracer, the output is saved as per mil.

In addition, atmospheric – atm_* and biological export – bio_fpart_* tracers are also supported.

Important. A DOT is required at the end of each output variable line, which helps the algorithm distinguish different tracers.

Note that because there is an ambiguity between atmosphere and ocean temperature, as they have the same name, when specifying any of atm, ocn, or sed tracers for saving, add 'atm', 'ocn', or 'sed' to the string. Use underscores ('_') to separate parts of the string. For now, all 'sed' tracers requested are assumed to be particulate fluxes. (Again, as before.)

Also note that for saving carbonate chemistry anywhere in the ocean and at any time (not just associated with time slice saving), you need to specify that the carbonate system is re-solved frequently. The setting for this is:

```
bg_ctrl_carbchemupdate_full=.true.
```

Note that there may be considerable computational overheads to this. However, the surface equilibrium is solved frequently by default, and if you only want surface, you do not need to add this setting. (So I guess it is only if you want benthic carbonate chem saved is there any issue.)

There is also a previous/old way of saving just orbital insolation:

For evaluating and checking orbital experiments in **cGENIE.muffin**, it is useful to be able to extract commonly used diagnostics of orbital variations in insolation, e.g. June 21 65N. **cGENIE.muffin** can be configured to save the *time-series* (at the standard save frequency for *time-series* output) at two different 'j' grid positions (latitude) – typically one Northern and one Southern hemisphere, and at specific points in the seasonal cycle. The relevant parameters are:

- **bg_par_sig_j_N** Sets the 'j' latitude grid point location for extracting the insolation. On a 18×18 grid, a value of 17 is as close as you are going to get to 65N.
- **bg_par_sig_j_S** As above, except for the Southern hemisphere (although there is nothing stopping you from choosing 2 Northern or 2 Southern hemisphere points, just what out for the automatic output column labelling that might cause confusion).
- **bg_par_t_sig_count_N** Sets the time-step in the **BIOGEM** tie-stepping cycle at which the insolation value is extracted. Care is needed here in distinguishing between the primary ocean (**GOLDSTEIN**) time-step, and **BIOGEM**, which is typically sub-stepped. For example, on an $18 \times 18 \times 8$ grid, the **GOLDSTEIN** time-step is 48 (per year), and **BIOGEM** time-steps every other **GOLDSTEIN** step, i.e. 24 per year. A value for this parameter of 12 is then approximately June 21.
- **bg_par_t_sig_count_N** As above, except for the Southern hemisphere.

The *time-series* output files is called: `biogem_series_misc_ocn_insol.res` (and has pretty self-explanatory columns).

Saving global/annual (insolation) changes

Two new misc category time-series files have been provided:

`biogem_series_misc_ocn_insol.res`

and

`biogem_series_misc_ocn_swflux.res`

with the the SW (shortwave) flux (`swflux`) being equivalent to the incident strength of solar radiation at the surface (`insol`) but accounting for the prescribed planetary albedo. Both variables are calculated and saved on a global mean ocean grid basis (2nd data column) and have units of W m⁻². In addition, to help diagnose orbital variability, `biogem_series_misc_ocn_insol.res` includes two further insolation variables (3rd and 4th columns). These reflect the strength of insolation at a single point in the annual cycle and at discrete latitudes (i.e. `j` grid indices). (The insolation at 2 different latitudes are saved so that both e.g. N and S hemisphere insolation signals can be simultaneously recorded.)

Three new namelist parameters are provided to configure this:

1. `bg_par_t_sig_count` – which sets the BIOGEM 'time-step' in the annual cycle at which the insolation value will be saved. e.g. for 96 time-steps in the ocean physics, and a 2:1 GOLDSTEIN:BIOGEM gearing (the default for the 16 level configuration), there are 48 BIOGEM time-steps. (It is left to the user to work out which part of the annual cycle *cGENIE* starts at (i.e. time-step #1) – I haven't a clue ...)
2. `bg_par_sig_j_N` – sets the '`j`' value for a northern hemisphere (but could be southern) snap-shot.
3. `bg_par_sig_j_S` – sets the '`j`' value for a southern hemisphere snap-shot.

By default `bg_par_sig_j_N` is assigned a value of 2 and `bg_par_sig_j_S` a value of 1 (on account of the maximum grid size not being *a priori* known). The default setting of `bg_par_t_sig_count` is 1 – i.e. the first (BIOGEM) time-step in the annual cycle.

21. Model Code

Best not to touch it... but

21.1 Testing

21.2**Common Modifications****Add 'name-list' (run time) parameters**

In order to create a new '*namelist*' parameter, i.e. a parameter whose value can be set in a *user-config* file, you need to edit a total of 4 files:

1. `*_lib.f90`

Add an entry in the relevant module library file, which for BIOGEM would be:

`biogem_lib.f90`¹.

The parameter must be defined (with an appropriate type) and added to the NAMELIST section at the top of the file. Simply follow the format of the existing entries and add to the most appropriate section of parameter categories. Note that the parameter name appears *twice* – one in defining its type, and once in adding the the parameter NAMELIST.

2. `*_data.f90` For completeness, there is an entry in the subroutine that reports the selected parameter options upon model start-up (if this reporting is selected in the first place ...), which for BIOGEM, is subroutine:

`sub_load_goin_biogem`

that lives in:

`biogem_data.f90`.

Again – simply follow the format of existing entries for creating a new one. Again: add in an appropriate section of parameter categories to prevent future coders going mad looking for something.

3. Add a new definition (including brief description and default value) in the xml definition file: `definition.xml`

that can be found in the directory:

`cgenie.muffin/genie-main/src/xml-config/xml` You will have to scroll down to find the section for the appropriate module, and then within that, the section for that category of parameter. Simply follow the format of the existing entries.

4. **<FILE>.f90 (or <FILE>.f77)**

Finally, edit into the appropriate FORTRAN source file the code that incorporates the parameter that you wish to use. D'uh!

Having ensured the model compiles and seems to do what you wish it do to – run a standard test to confirm that nothing obvious has been unintentionally affected by the change. To do this, simply type `make testbiogem` and confirm that the test passes ...

Add additional results output

If a new data field can be derived from an existing field, then creating additional results saving is relatively straightforward because no new time-averaging has to be carried out (i.e. you create the new field based on annual (or sub-annual) averages that are already calculated and available. (If not – refer to the 'full' data saving sub-sections.) Note that new tracers are automatically saved.

Data saving 101

For new *time-slice* saving, code needs to be added to either²:

`sub_save_netCDF_2d_USER`

or:

¹`cgenie.muffin/genie-biogem/src/fortran`

²`biogem_data.netCF.f90`

sub_save_netCDF_3d_USER

depending on whether the field is 2D or 3D, respectively. Add the code to the end of the subroutine (as marked). Follow the format of previous data saving as far as possible. For general format is:

- Add a conditional to define under what circumstances, particularly selected save options, the data is saved. DO NOT save it by default ... unless it is of vital importance to the future of the planet. Refer to the user-manual for the categories of save options, investigate what options are used for similar data fields, and use your common sense ...
- Specific a units name for `loc_unitsname`, or set to 'n/a' if not applicable (or non-dimensional).
- Initialize (zero) the local 2D (`loc_ij`) or 3D (`loc_ijk`) (depending on the data field) data array.
- Calculate the data, employing a nested loop if necessary (i.e. simply matrix math cannot be employed) and assign to the local data array.
- Add a call to `sub_adddef_netcdf` (don't ask questions – follow the general format).
- Add a call to `sub_putvar2d` (ditto).

For new *time-series* saving ... it is sort of both more and less complicated :o) The new code goes in `biogem_data_ascii.f90` but in two different places: `sub_init_data_save_runtime` and `sub_data_save_runtime`. For former creates the (ASCII) file and adds a header (to define the columns), and then closes the file. The latter opens the now existing file, writes the output, including the (*time-series* save point) time, and the closes the file. The code needed (again – follow the generla format and best to add the new code to the end of the subroutines) is hence ...

In `sub_init_data_save_runtime`:

- Add a conditional to define under what circumstances, particularly selected save options, the data is saved. DO NOT save it by default ... unless it is of vital importance to the future of the planet. Refer to the user-manual for the categories of save options, investigate what options are used for similar data fields, and use your common sense ...
- (If multiple different variables stored in or based on the same array, set up a loop.)
- Create the filename to be sued (`loc_filename`) via a call to `fun_data_timeseries_filename`. All files are `*_series` with the specific variable type and variable after, or if not a specific variable type, then `_misc`.
- Create the header text. If you start with a '%' then it is all the more MUTLAB friendly.
- Follow the sequence format of: CHECK / OPEN / CHECK / WRITE / CHECK / CLOSE / CHECK (yawn) ...

In `sub_data_save_runtime`:

- Add the same conditional as used in `sub_init_data_save_runtime`.
- (If multiple different variables stored in or based on the same array, set up a loop.)
- Construct the local filename `*exactly*` as before (`sub_data_save_runtime`) or it will not find the fiel you have created ...
- Calculate the data value(s) (`loc_sig`).
- CHECK / OPEN / CHECK / WRITE / CHECK / CLOSE / CHECK (yawn) ...

21.3 Uncommon (Ill-advised) Modifications

Define a new tracer

You probably should not be doing this ... but ... just in case ...

Basic definition procedure

1. The starting point to adding a new tracer in *cGENIE* is to add its definition to the relevant tracer definition file.

There are three tracer definition files that live in `cgenie.muffin/genie-main/data/input`:

- `tracer_define.atm`
- `tracer_define.ocn`
- `tracer_define.sed`

that house a list of atmospheric (gaseous), oceanic (dissolved), and sediment (solid) tracer definitions. Each file has a similar format, with a series of columns³ holding information on:

#01 The short (mnemonic) name for the tracer. This is used in creating the output filenames and netCDF variable names. In theory, this could be anything you like, but to a limit of 16 characters.

#02 The identifier (index) of the tracer. This must be a unique number – one number for each tracer.

#03 The 'dependency' of the tracer. For instance, an isotope depends on the bulk (and lower mass) tracer. A scavenged element depends on bulk organic matter. For a bulk tracer⁴, its dependency is itself.

The dependency is used in the code to automatically determine any tracer that it depends on and in the case of isotopes, to calculate the δ value.

#04 Is the tracer variable 'type', used internally to determine what to 'do' with a specific tracer:

- '1' → assigned to primary biogenic phases; POM (represented by POC), CaCO₃, opal (all contributing to bulk composition)
- '2' → assigned to abiotic material (contributing to bulk composition); det, ash
- '3' → assigned to elemental components associated with POC; P, N, Cd, Fe
- '4' → assigned to elemental components associated with CaCO₃; Cd
- '5' → assigned to elemental components associated with opal; Ge
- '5' → assigned to elemental components associated with det; Li
- '7' → assigned to particle-reactive scavenged elements; 231Pa, 230Th, Fe
- '8' → assigned to carbonate 'age'
- '9' → assigned to the fractional partitioning of biogenic material (for remineralization purposes)
- '10' → assigned to misc / 'inert'
- '>10' → assigned to isotopic properties: 11==13C, 12==14C, 13==18O, 14==15N, 15==34S, 16==30Si, 17==114Cd, 18==7Li, 19==144Nd, 20==44Ca, 21==98Mo, 22==56Fe.

#05 Tracer long name. Cannot be more than 128 characters in length.

#06 Units.

#07 Minimum valid value for the tracer. Values lower than this are classed as NaN in the

³The meaning of the columns is also summarized at the end of the file.

⁴The elemental components of organic matter (P, N, etc) count as bulk tracers for the purpose of dependency.

netCDF output.

#08 Maximum valid value. Values higher than this are classed as NaN in the netCDF output. Hence, the first thing to do is to add an entry for the required tracer(s) to the relevant file(s), using the above information and keeping a consistent format and convention with the existing tracers.

If you have an elemental or isotopic tracer that is taken up by a growing phytoplankton cell and incorporated structurally into POM, you will also need to define equivalent dissolved (and recalcitrant) dissolved organic matter tracers (DOM and RDOM). If the tracer is associated with organic matter (or other particles), then you require a scavenged particulate tracer but no corresponding dissolved tracer. For something like iron, which is both incorporated into the cell, and scavenged, you need both types of particulate tracer plus a set of dissolved organic matter tracers ...

2. A set of tracer-related definitions also exists in:

`cgenie.muffin/genie-main/src/fortran/cmngem/gem_cmn.f90`

Requiring some editing-attention here is⁵:

ca. L24-26 Three parameters define the main tracer array sizes. (Unfortunately this information has to be duplicated elsewhere due to the peculiarities of the GENIE code structure – see below.) Their values must be equal to the total number of tracers defined in the tracer definition files.

ca. L72-261 So as to simplify referencing the tracers in the code, each tracer is assigned a simple mnemonic. This mnemonic may or may not be the same as the short name defined in the tracer definition files. With a little bit of code, the tracer definition short name could have been turned into an index value, but to date this has not been implemented. However, while slightly tedious to set up, created fixed and compile-time rather than run-time mnemonic assignments is going to be somewhat faster as well as making the code slightly more compact.

ca. L630-642 Definition of the isotope standards. Only if creating a new isotope system does this need to be edited.

So effectively, just the total number of tracers, and the addition of the tracer mnemonic name, has to be edited in this file.

3. An extremely unfortunate fact of GENIE code structure life is that the total tracer numbers are defined a second time in:

`cgenie.muffin/genie-main/genie_control.f90`

at ca. L144-146, and need to be edited consistent with the values in `gem_cmn.f90` (see above). Why ... ? Please don't ask.

4. Equally, or arguably even more annoying and opaquely justified, is the need to edit a number of entries in:

`cgenie.muffin/genie-main/src/xml-config/xml/definition.xml`

A number of sets of runtime parameters exist with one entry per tracer. Given that all runtime parameters must be defined in the xml definition file, means that every time a new tracer is created, a number of xml entries must be created⁶ :(The approximate line occurrence of this outrage against all good programming practice, are as follows:

ca. L397-1159 The tracer arrays holding information about which tracers are selected.

ca. L2538-3271 Tracer arrays for the initial value of ocean (dissolved) tracers, as well as a tracer value modification array.

⁵Watch out that the line numbers may have changed somewhat ...

⁶Note that there are no sediment (solid) tracer arrays for either initial composition or forcings.

ca. L4343-5234 Tracer arrays for atmospheric (gaseous) and ocean (dissolved) tracers governing tracer forcings (value and tie-scale).

ca. L5292-5370 Tracer array for the initial value of atmospheric (gaseous) tracers.

To edit in new entries – simply follow the format of the last entry.

In addition – at the start of each array definition (xml tag starting <ParamArray>), the array size is defined (`extent=`). This also needs to be edited to reflect the inclusion of additional tracers.

5. Even more annoying, if possible, is the need to then edit the array entries in the Python xml parameter translation script:

```
cgenie.muffin/genie-main/config2xml.py
```

It should be obvious where there additional entries corresponding to the new tracers need to be added. Just be careful of the formatting (the last entry in the list of tracers for each array not having a ',' terminating the line. Otherwise, simply follow the existing format.

That is it for the basic tracer definition. The model should now compile without error (and still pass its `make testbiogem`) although you'll need to do a `make cleanall` first. However, whilst 'knowing' about the possibility of the new tracers, at this point you have not actually selected any for an experiment and no initial conditions will be read in, nor relevant output created.

Testing

Having checked the model compiles and passes the basic BIOGEM test, the next step is to check that the new tracer(s) is initialized correctly (atm and ocn tracers) and that appropriate output is generated.

1. Create a new *base-config* by taking an appropriate/comparable existing *base-config*⁷ and modifying it.

The first modification is to increase the number of ocean tracers, defined on the line headed by:

```
# Set number of tracers
```

incrementing the total by the number of additional tracers to be included in the new *base-config* as compared to the original one.

The second modification is to select the additional tracers – simply add appropriate entries to the list of selected tracers (the tracers are selected by setting the parameter value to `.true.` as by default they are `.false.`).

The final modification, in the case of atmospheric (gaseous) and ocean (dissolved) tracers is to set an initial value. If you do not do this, by default, concentrations are initialized to zero.

2. Now go create a *user-config* file. Copy an EXAMPLE config file – one corresponding to the unmodified *base-config*, if possible. You should rename it, and although no modifications are required to the parameter settings in order for the model to run, to ensure all possible output is produced, set:

```
bg_par_data_save_level=99
```

3. Run a brief experiment and check that the tracer appears in the output – both time-series and netCDF. For ocean tracers, the concentration field should progressively look like salinity as concentration changes at the surface will be influenced by P-E. (Remember that at this point, no other transformations or changes of tracer have been defined – just that there is a tracer and it is initialized to a certain value.)

⁷/cgenie.muffin/genie-main/configs

Adding (and testing) a basic tracer biogeochemical cycle

Now for the trickier part, assuming you do not just want a simple passive tracer (there are plenty of 'color' tracers defined already!) and assuming you have got the tracer already successfully configured and running as a simple passive tracer (i.e. previous steps).

The example will be for an ocean tracer that is incorporated into particulate organic matter (POM) (and hence creating an associated sedimentary tracer) during biological productivity. Other and much more fun and entertaining complexities will apply if e.g. the ocean tracer exchanges with the atmosphere and hence is associated with an atmospheric tracer.

1. The relationships between different sorts of tracers, e.g. dissolved and gaseous, and dissolved and solid, are defined in subroutine `sub_def_tracerrelationships`⁸

If there is an equivalent dissolved organic matter tracer corresponding to the particulate organic matter one, the relationship between POM and DOM (and also RDOM) also needs to be defined.

Typically, the relationship between a particulate and dissolved inorganic, or particulate and dissolved organic will be 1.0, but depending on the species concerned, it may be 2.0 (or its reciprocal) and/or negative. These values can be modified later if necessary, and this occurs depending on the redox state of the ocean in⁹:

`sub_data_update_tracerrelationships`

2. For a dissolved inorganic species being taken up biologically, a 'Redfield' like ratio is defined and used to relate the cellular quotient of the tracer versus carbon¹⁰. An array (`bio_part_red`) stores the relationship of every particulate tracer to every other particulate tracer. It is hence mostly zeros, except for the ratios of the particulate tracers to carbon in both organic matter and CaCO_3 (and for that matter, opal)) and of the isotope ratios of species to their bulk equivalent. The array is (re)populated each time-step in `sub_calc_bio_uptake`¹¹, typically by directly applying a globally applicable ratio that is read in at run-time (and hence requiring a new *namelist* parameter to be defined), by some function of ambient environmental conditions that is often a modification of the run-time parameter.

If steps #1 and #2 have been completed correctly, there should now be a biological cycle of the new tracer, with it being taken up at the ocean surface and incorporated into POM with a specific ratio compared to carbon (set by the new *namelist* parameter) and resulting in depletion of the inorganic dissolved tracer at the ocean surface. Conversely, there should be elevated concentrations of the inorganic tracer at depth, mirroring the pattern of e.g. $[\text{PO}_4]$. Creation and subsequent remineralization of a corresponding tracer incorporated into DOM (and RDOM if selected) should occur automatically. However, the recommended first step in testing the newly defined biogeochemical cycle is to disable all DOM formation, by setting:

`bg_par_bio_red_DOMfrac=0.0`

Also recommended is to enable 'auditing' of all the tracer inventories to ensure that tracers are not being spuriously created or destroyed by:

`bg_ctrl_audit=.true.`

⁸`cgenie.muffin/genie-main/src/fortran/cmngem/gem_util.f90`

⁹`cgenie.muffin/genie-biogem/src/fortran/biogem_data.f90`

¹⁰A carbon currency is used in the model rather than phosphate, despite the classic Redfield ratio being defined relative to a phosphate quotient of 1.0

¹¹`cgenie.muffin/genie-biogem/src/fortran/biogem_box.f90`

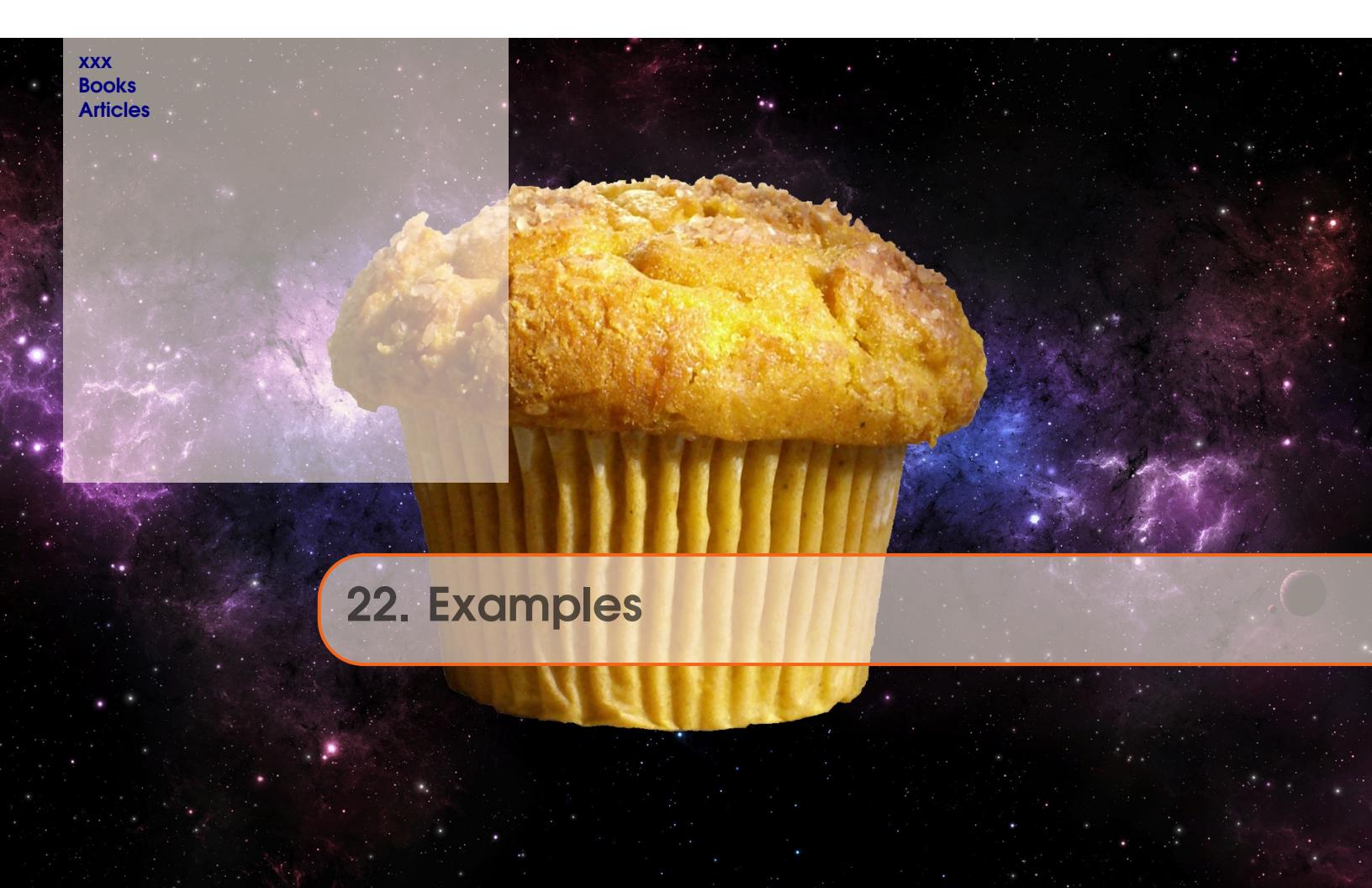
Scavenged tracers are automatically remineralized along with the corresponding parent particulate tracer by default¹². However, there is no scavenging or creation of scavenged tracers by default. A call would need to be added to¹³:

```
sub_box_remin_part
```

(ca. L2961), e.g. following the examples of Fe scavenging and H₂S reaction with organic matter (treated as a form of 'scavenging' for simplicity of code structure), plus a corresponding subroutine added in which is the scavenged particulate tracer concentration is calculated and the removal of the corresponding dissolved tracers set.

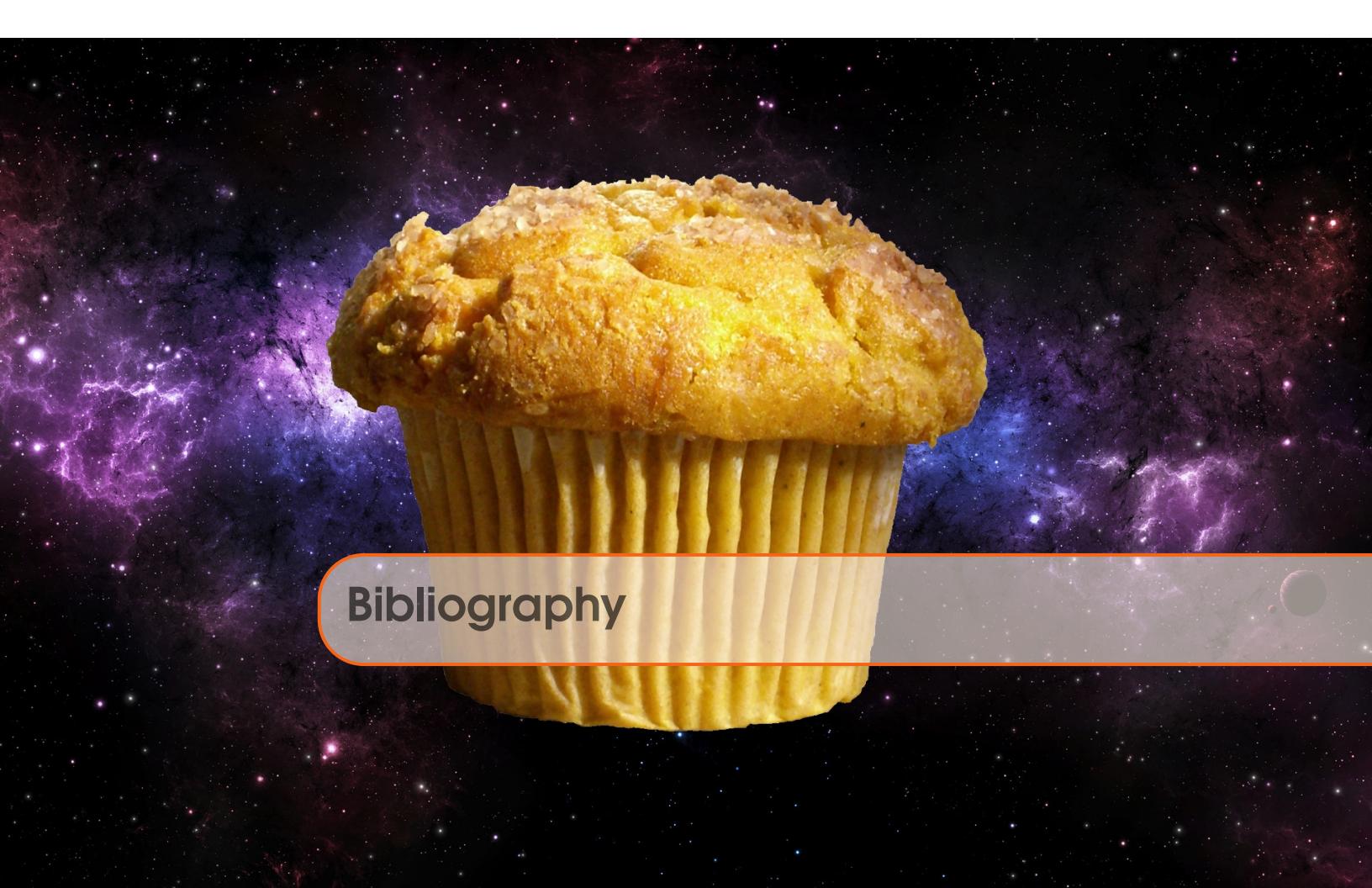
¹²They can instead be set to remain in the scavenging medium by setting a non zero value of `par_scav_fremin`, which sets the fraction of the scavenged tracer that is remineralized along with the degraded parent particulate tracer.

¹³`cgenie.muffin/genie-biogem/src/fortran/biogem_box.f90`



22. Examples

22.1 **xxx**



Bibliography

Books

Articles

