User manual for cGENIE: 'muffin' pre-release version

Andy Ridgwell

February 17, 2015

Contents

1 READ-ME blurb

This document covers the directory structure of the source code and ancillary data input files, instructions for compiling the *c*GENIE executable, and how to configure and execute an experiment (i.e., how to get some actual work done). Also covered is how to specify the results you want saved and at what points in time, in what format they are saved, and (some ways of) getting the results visualized. This is part of a series of documentation to help you get started using cGENIE, including:

1. **Quick Start Guide** (`cGENIE.QuickStartGuide.pdf`) – Abbreviated instructions for obtaining, configuring, and testing the model.
2. **READ-ME** (`cGENIE.muffin.README.pdf`) – Miscellaneous notifications including recent changes to code and model behavior.
3. **Examples** (`cGENIE.muffin.Examples.pdf`) – Various example experiments, including example model configurations, experimental designs, and associated parameter descriptions.
4. **HOW-TO** (`cGENIE.muffin.HOWTO.tex`) – Potted explanations of how to get useful stuff done, e.g., spinning up deep-sea sediment distributions.
5. **User manua**l (`cGENIE.muffin.User_manual.pdf`) (*this document*)

If you favor the jumping-in-with-both-feet-blind approach, then just read the **Quick Start Guide**, but it is advisable to read at the very least the **READ-ME** as well (have as guess as to why it is call that). Scan the FAQ section in the **User manual**, lest any of the FAQs become EMFAQs (Even More Frequently Asked Questions) as well as the topics covered in the **HOW-TO** document. Finally, additional information as to the practicalities of setting up and configuring experiments can be gleaned from **Examples**, which contains a number of described example configurations of the model and experiments including a number that have been published and hence the resuts of ideally, reproduceable.

The LATEX source for all the documents resides in the `cgenie.muffin/genie-docs` directory and from which both postscript (.ps) and Adobe (.pdf) format versions can be built. This can be done either by some devious piece of LATEX software (e.g. TeXnicCenter for Windoz) or can be built under linux at the command line by:

- `$ make cgenie-user-manual` – builds the **User manual**.
- `$ make cgenie-examples` – builds the **EXAMPLES** documentation.
- `$ make cgenie-howto` – builds the **HOW-TO** documentation).

and filespace cleaned up by:

- `$ make clean-cgenie` – cleans all built files associated with the `cgenie.muffin.*` documentation.

Pre-built PDF versions of all available official documentation[1] are provided on the *c*GENIE homepage under `cGENIE resources: DOCUMENTATION`.

PLEASE edit and update the documentation and populate it with wonderful, useful, and hopefully factually accurate and typo-free things. Spread the love :o)

1.1 Naming conventions blurb

The original modular Earth system model (of intermediate complexity) was named GENIE, which stands for: Grid ENabled Integrated Earth system model[2]. There were originally 2 primary variants - one with a simple (2-D, EMBM - see below) atmosphere which was referred to as 'GENIE-1', and one which employed a 3-D dynamical GCM atmosphere, called 'GENIE-2'. However, much of the code for the dynamical atmosphere and ice sheet components was pretty dodgy and modern FORTRAN compilers increasingly refused to compile the stuff. There were also a large number of modules and variants of that hardly anypony used. So I deleted them all. And then the sun came out. *c*GENIE is a

[1] Note that there is no guarantee that all the documentation is entirely up-to-date or consistent with the current state of the model code, and built PDF files may lag behind the LATEX version (so better to build your own).

[2] One of the World's less sane acronyms.

carbon cycle centric variant of GENIE-1 and in terms of code history, is a branch developed off of the primary model development pathway (trunk) of GENIE. The current pre-release version of *c*GENIE is code-named **muffin**[3].

It will be assumed that the code directories (`cgenie.muffin/`), plus output (results), archive, and log (more of this later) directories, all reside in a base directory `$MUFFINHOME`[4], which is typically, but not always, your home directory and is where the source code was installed (`$HOME` by default).

The names of the science modules currently available are listed in Appendix A (page 38).

1.2 Model background blurb

The core model component is the climate model C-GOLDSTEIN, as described in *Edwards and Marsh* [2005] (calibrated in *Hargreaves et al.* [2005] with minor alteration (to the equation of state) as described in *Ridgwell et al.* [2007a]). It comprises three modules:

- GOLDSTEIN (the 3-D ocean circulation model of *Edwards and Shepherd* [2001])
- dynamic-thermodynamic sea-ice, and
- 2-D EMBM (Energy-Moisture Balance Model) following *Weaver et al.* [2001].

In GENIE terminology, the EMBM atmosphere plus GOLDSTEIN ocean plus sea-ice (linked to GOLDSTEIN) has the '*flavor*' (i.e., specific combination of science modules):

`cgenie.eb_go_gs`

The addition of ocean (and atmosphere) biogeochemistry to C-GOLDSTEIN gives you a coupled climate - carbon cycle model which was called CB-GOLDSTEIN in its non-modular, pre-GENIE incarnation. CB-GOLDSTEIN has an ocean biogeochemistry module called BIOGEM (for 'BIOGEochemistry Model') and a module to update and keep track of the atmospheric 'chemistry' (greenhouse gas concentration and isotopic composition) called ATCHEM ('ATmospheric CHEmistry Model')[5]. A simple coupler also provides BIOGEM with information about the overlying atmospheric composition, and if required, can drive the EMBM with trace gas concentrations for calculating radiative forcing of climate. As well as indirect coupling via the atmosphere and EMBM, BIOGEM and GOLDSTEIN are directly coupled through the exchange of 3-D array information regarding the geochemical composition of the ocean. BIOGEM also makes copies of various key configuration parameters at start-up as well as paying attention to sea-ice extent etc, and provides these all as output (independently of GOLDSTEIN). The *flavor* equivalent to CB-GOLDSTEIN is:

`cgenie.eb_go_gs_ac_bg`

Regardless of whether or not a biogeochemistry component is included, for the purpose of this manual, the model framework is now called *c*GENIE (muffin release). Indeed, because the ocean biogeochemsity module provides a variety of data pre-processing and saving capabilities, it tends to be included even when the only ocean tracers are temperature (T) and salinity(S). So even for a climate-only experiments, the *flavor* employed is invariably `cgenie.eb_go_gs_ac_bg`.

The configuration, calibration, and evaluation of *c*GENIE (which back then was called GENIE-1), is described in *Ridgwell et al.* [2007a]. Earlier incarnations of the ocean biogeochemistry are described in *Cameron et al.* [2005] and *Lenton et al.* [2006] (and used CB-GOLDSTEIN). So it should be perfectly clear now – the model that was formally called GENIE-1 is now known as *c*GENIE. Previously published references to *c*GENIE were sometimes really GENIE-1. Except when they were CB-GOLDSTEIN.

The global carbon cycle can be further extended with the addition of a representation of the mass balance of solid material deposited to deep-ocean sediments - a module called SEDGEM ('SEDiment GEochemistry Model')[6]. This is coupled via an interfacing module to BIOGEM. The overall coupled climate-ocean biogeochemistry-sediment geochemistry model was previously termed CBS-GOLDSTEIN, but in GENIE terminology the *flavor* is:

`cgenie.eb_go_gs_ac_bg_sg`

[3] I feel a baked goods products major version naming theme coming on ... move over Andriod.
[4] It is possible I accidently slip into using $HOME or ~ to designate this directory, so please stay awake!
[5] Sadly, the acronyms are getting no better ...
[6] It just keeps getting tackier and tackier.

The configuration, calibration, evaluation, and application of *c*GENIE including deep-sea sediments, is described in *Ridgwell and Hargreaves* [2007] and *Ridgwell et al.* [2007b]. Further developments to the sediment model are described in *Ridgwell* [2007].

Finally, when allowing burial of biogenic material in deep-sea sediments using the SEDGEM module, a further module is available to provide the (long-term) balance the loss of matter from the ocean. This is called ROKGEM (for: 'ROcK GEochemistry Model') and calculates the solute supply to the coastal ocean resulting from the weathering on land of exposed rock surfaces and soil minerals. The *flavor* of *c*GENIE incorporating the weathering module is then termed:

`cgenie.eb_go_gs_ac_bg_sg_rg`

(A final module in this series that is used in conjunction with SEDGEM and ROKGEM to accelerate the long-term weathering minus sedimentation mass balance, is called GEMlite, with mnemonic: `gl`.)

Appendix A summarizes the different science modules in *c*GENIE.

Suggested background readings to help understanding the basic climate model, its typical applications, its successes and perhaps more importantly its limitations are:

- *Edwards and Shepherd* [2001] – the ocean circulation model component
- *Edwards and Marsh* [2005] and *Marsh et al.* [2009] (for the fast climate model core component, C-GOLDSTEIN)
- *Hargreaves et al.* [2004], *Annan et al.* [2005] – parameter tuning and uncertainty

For atmosphere-ocean(-sediment) carbon cycling and biogeochemistry, try:

- *Cameron et al.* [2005] – for an earlier representation of biogeochemical cycling in the ocean and atmosphere
- *Ridgwell et al.* [2007a,b] – basic ocean-only biogeochemical cycling and parameter tuning
- *Ridgwell and Hargreaves* [2007], *Ridgwell* [2007], *Kirtland Turner and Ridgwell* [2013] – description and tuning of the sediment model
- *Colbourn et al.* [2013] – description of the terrestrial weathering model

Versions of most of the relevant model references can either be downloaded from my publications server.

Also refer to the companion documentation to this manual.

2 Getting your grubby little mitts on (and updating) the model

2.1 cGENIE software versioning and revision control

First, you will need an 'SVN' compatible software versioning and revision control client of some sort installed on the machine you want to run the model on (or at least, on the machine you want your local copy of the source code to be maintained on), e.g. see the Wikipedia entry for a basic over-view of obtaining the source code. In summary:

If you are using linux, then you will most likely have the command line client already installed. Try:

```
$ which svn
```

to see if the client is in your path. Command line and GUI clients for SVN are available for just about any operating system. `tortoisesvn` is popular for windows. `SVNx` is a good choice for the Mac[7].

To get a (read-only) copy of the current muffin branch of *c*GENIE source code – from your home directory (or elsewhere, but several path variables will have to be edited – see below), type:

```
svn co https://svn.ggy.bris.ac.uk/subversion/genie/branches/cgenie.muffin
--username=genie-user cgenie.muffin
```

for the 'head' (current development version). NOTE: All this must be typed continuously on ONE LINE, with a S P A C E before '`--username`', and before '`cgenie`'. Unless you have logged onto the `svn` server before from your computing account, you be asked for a password – it is `g3n1e-user`.

Once installed, to update an existing checkout, you *do not* need to delete the current installation and re-install – instead: simply `cd` to the *c*GENIE installation directory (e.g. `$MUFFINHOME/cgenie.muffin` and issue the update command:

```
$ svn update
```

note that you could update just a sub-tree (here the BIOGEM module) of your checkout by changing directory to

```
$MUFFINHOME/cgenie.muffin/genie-biogem
```

and then issuing `svn update`. Or you could just update a single file with `svn update` *`filename`* from within the appropriate directory, or by additionally supplying the full path.

2.2 Configuring cGENIE for your local software environment

You'll need to have installed or linked to an appropriate FORTRAN compiler and netCDF library[8].

2.2.1 FORTRAN compiler

- `gfortran`
  For the GNU FORTRAN compiler (`gfortran`) – **version 4.4.4** or later is recommended.
- `ifort`
  For the Intel FORTRAN compiler (`ifort`) – take care with setting the necessary compiler environment variables. A script is provided for setting these as part of the Intel compiler installation, called `ifortvars.sh` (`find` it if in doubt as to where it is). This can be e.g. added to the `.bashrc` script:
  `./opt/intel/composer_xe_2015.0.090/bin/ifortvars.sh intel64`

[7] I don't have a Mac, I wouldn't know – I am just making this stuff up as I go along.
[8] If running using an account on one of the Bristol clusters – this is all configured for you.

2.2.2 netCDF

Currently, the default compilation PATHS require that netCDF is **version 4.0** – this is one of the last versions in which C and FORTRAN libraries were combined. To use the current netCDF – install the `C` and then the `FORTRAN` libraries (refer to the netCDF documentation. Edit the PATHS as marked in `makefile.arc`[9]. (I have not actually tried this myself – hopefully it works! :o))

Regardless, it is important that the netCDF library is build with the *same* compiler as you intend to compile *c*GENIE with.

2.2.3 Environment variables

You need to set a couple of environment variables – the compiler name, netCDF library name, and netCDF path[10]. These are specified in the file `user.mak` (`genie-main` directory). If the *c*genie 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) ...

There are also alternative options for how *c*GENIE is compiled, particularly in respect of the degree of optimization and whether or not to include debug information (and what level of detail). in `user.mak`, alongside a flag to select which FORTRAN (`F77=gfortran` is the default) and C compilers (`CC=gcc`, `CXX=g++`) are used, are options for the 'build type' (only un-comment (`#`) one):

1. `BUILD=NORMAL` – build *c*GENIE with a moderate degree of optimization
2. `BUILD=SHIP` – build *c*GENIE with the highest/most aggressive degree of optimization **[default]**
3. `BUILD=DEBUG` – build *c*GENIE with full de-bugging (warning: slow!)
4. `BUILD=PROFILE` – (blah other debugging options blah)
5. `BUILD=BOUNDS` – (blah other debugging options blah)

Be aware that the compiler flags have not been optimized or adjusted for some years, especially true of the non `gfortran` options. For each options, the selection of compiler flags are defined in `makefile.arc` (in genie-main) and can be edited if necessary.

2.2.4 Verifying the code

To test the code installation – change directory to `cgenie.muffin/genie-main` and type:

```
make testbiogem
```

This compiles a carbon cycle enabled configuration of *c*GENIE 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
```

[9] `cgenie.muffin/genie-main`

[10] If running using an account on one of the Bristol clusters – the relevant netCDF path for each cluster appears (commented out) at the bottom of the file – ensure that the appropriate value of the `NETCDF_DIR` environment variable is not commented out (and the others are).

3 Running cGENIE

I'll discuss this in the reverse order just for the hell of it – setting a model experiment going *first* ... and only then describing what you should have done in the first place to have set it up correctly ;)

It is important to recognize from the onset that there are two basic modes of operating *c*GENIE - as a *new* or *continuing* run (experiment). A *new* run is one which starts off from a default state of the system, which for the various science modules in *c*GENIE is:.

- For the ocean model GOLDSTEIN, this is with the ocean set to uniform temperature and salinity throughout[11], and no momentum.
- In BIOGEM, the dissolved tracers are initialized with default values applied to all wet cells while all particulate concentrations in the ocean are zero.
- In ATCHEM all tracer gas concentrations (and isotopic properties) are initialized with uniform default values everywhere in the atmospheric grid.
- In SEDGEM the sediments are set to a homogeneous (detrital or ash tracer) composition throughout.

A *continuing* run will carry on from where-ever a previous run left off, using a set of files called *restart* files that between them completely define a snapshot state of the system. More on this (plus examples) later (and in the Examples document).

A model experiment, whether *new* or *continuing*, can be initiated in two different ways ...

3.1 ... manually ... :(

The default configuration (whatever that is) of cGENIE can be run from `$MUFFINHOME/cgenie.muffin/genie-main` by typing:

```
./genie_example.job
```

`genie_example.job` is a shell script which sets up the default model. An alternative setup of the *c*GENIE is enacted by specifying a configuration file that contains changes compared to the built-in defaults:

```
./genie_example.job -f anon.config
```

where `anon.config` is a specified model configuration `.config file`.

Since the Great Cull of legacy GENIE code and supporting files, *c*GENIE comes with a only a few pre-prepared configurations files of this sort. These live in `cgenie.muffin/genie-main/configs` and their file names are typically structured in the form `genie_aa_oo_ss_xxxx.config`. In fact, all this is pretty much redundant now. However, when an experiment finishes, the relevant `anon.config` file together with a copy of `genie_example.job` is saved in the `archive` sub-directory of the results directory meaning that the experiment can be re-run as per the command above[12].

There are also extremely fancy ways of running the model using `xml` formal configuration files, but I have no truck with any of this. Work it out for yourself and give it a try if you want. Except I might have already deleted the more fancy of the possibilities.

3.2 ... via a shell script :)

For configuring and running experiments interactively or submitting jobs to a cluster queue, an executable shell script called `runmuffin.sh` (similar to the `runcgenie.sh` and `old_rungenie.sh` variants that preceded it). By default, it configures the model with 96 time steps per year compared with 100 previously. There is hence a varient that assumes a 100 time steps per year – `runmuffin.t100.sh`. Some degree of guidance as to when is appropriate to use which (generally, old published configurations used 100 time steps per year) can be found by leafing through the **Examples**

[11] See **HOWTO** for details of how to initialize with non-uniform initial T and S distributions.
[12] A copy of the compiled model executable is saved in the main results directory.

document. `runmuffin.sh` resides in the directory `cgenie.muffin/genie-main`. Note that `runmuffin.sh` **MUST** have executable permissions (`chmod u+x runmuffin.sh`).

You use `runmuffin.sh` to run *c*GENIE, by invoking[13]:

```
$ ./runmuffin.sh <options>
```

Now pay attention: there are 4 (FOUR) parameters that **must** be passed to the `runmuffin.sh` script as `<options>`. Two of these options point to files of parameter values that together completely define your experiment – for convenience, the list of parameters is divided into 2 different files – the *base-config* and the *user-config* file. The other options define the directory where the *user-config* file can be found, and the run duration. Taking as an example a published experimental design used by *Cao et al.* [2009], these options would be:

1. The 1^{st} passed parameter is the name of the file[14] defining the basic configuration of the model for the experiment, including the combination of modules selected (the *flavor*), continental configuration and resolution, and selected tracers. This file is termed the *base-config*. In this example it would be:

   `cgenie.eb_go_gs_ac_bg.worjh2.ANTH`

   Rarely will you have need to edit or create a new *base-config* file and often a published configuration will already exist and new experiments created by including parameter modifications in the *user-config* file.
2. The 2^{nd} passed parameter is the path to the *user-config* file (the file defining the specific experiment), relative to the director `cgenie.muffin/genie-userconfigs`. In this particular example, this would be:

   `/`

   because the *user-config* `EXAMPLE.worjh2.Caoetal2009.SPIN` lives in `cgenie.muffin/genie-userconfigs`. However, this parameter might well be a sub-directory (or sub-sub-directory) of `cgenie.muffin/genie-userconfigs` for the convenience of grouping related experiments in their own sub-directory.
3. The 3^{rd} passed parameter is the name of the *user-config* file[15] containing the definition of the specific experiment to be run, here:

   `EXAMPLE.worjh2.Caoetal2009.SPIN`
4. The 4^{th} passed parameter is the length of the experiment **in integer years**[16].

A typical complete command line to run the model might thus look something like:

```
./runmuffin.sh cgenie.eb_go_gs_ac_bg.worjh2.ANTH / EXAMPLE.worjh2.Caoetal2009.SPIN 10000
```

(**taking care that it is all on one line and additionally not omitting any of the S P A C E S**)

There is also an optional 5^{th} parameter that can be passed:

5. The name of any *restart* required[17].

If there is no 5^{th} parameter then cGENIE will run from cold (i.e., with no *restart*). If the 5^{th} parameter is present then *c*GENIE will attempt to run from a previously generated *restart* state – an experiment with the name as specified by the parameter..

A typical command line to run the model using a specified *restart* might thus look something like (all one line):

```
./runmuffin.sh cgenie.eb_go_gs_ac_bg.worjh2.ANTH
 / EXAMPLE.worjh2.Caoetal2009.historical 2000 cgenie.eb_go_gs_ac_bg.worjh2.ANTH
```

in which the experiment `EXAMPLE.worjh2.Caoetal2009.historical` will be run for 2000 years, using the *base-config* specified by `cgenie.eb_go_gs_ac_bg.worjh2.ANTH`, and using as a *restart* (starting point) the experiment saved in this location:

```
~/genie_output/EXAMPLE.worjh2.Caoetal2009.SPIN
```

Full details regarding what `runmuffin.sh` actually does can be found in Appendix D (page 44).

[13] (from `cgenie.muffin/genie-main`)
[14] Omitting the '`.config`' bit from the filename.
[15] This is also the name that will be used to create the associated experiment results directory.
[16] This must be entered on the command line as an integer, even though GENIE will be treating it as a real.
[17] There must exist a subdirectory in `~/cgenie_output` with this name.

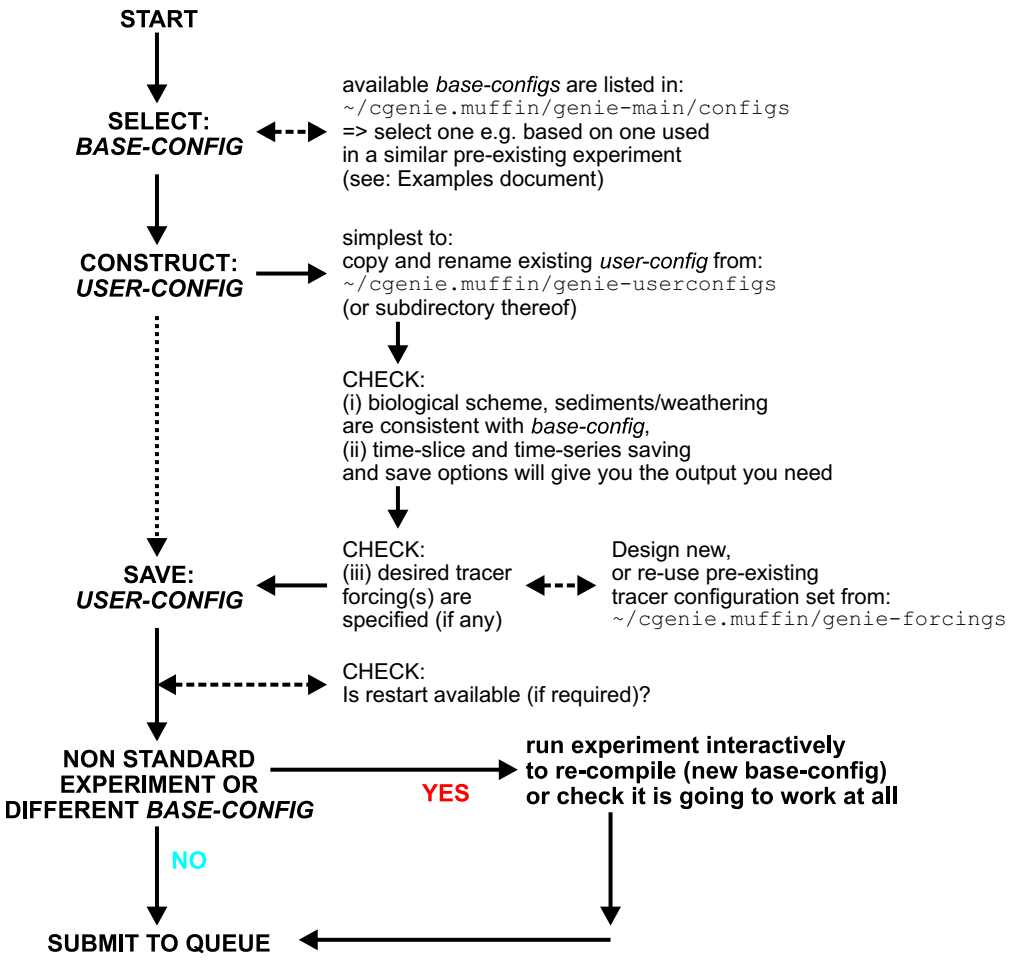

Figure 1: **Schematic of the sequence-of-events in configuring and running an experiment.**

3.3 Job submission to a cluster

How you do this depends somewhat on the cluster. For a University of Bristol cluster, a typical command would be[18]:

```
$ qsub -j y -o cgenie_log -V -S /bin/bash runmuffin.sh
        cgenie.eb_go_gs_ac_bg.worjh2.ANTH / EXAMPLE.worjh2.Caoetal2009.SPIN 10000
```

You can check the status of the SGE job queue with the command:

```
$ qstat -f
```

and you can kill a job with the `qdel` command, the job numbers being given by the `qstat` command.

NOTE: It may be that the FORTRAN compiler is not accessible by the computer nodes. The implication of this is that **the *c*GENIE 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.

[18] (issued from `cgenie.muffin/genie-main`)

4 Specifying time-dependent changes to tracer concentrations ('forcings')

Even the most highly coupled instance of GENIE is far from being a complete description of the Earth system. For instance, the lack of a dynamic atmosphere able to perform tracer advection means that dust deposition to the ocean surface cannot be internally calculated. With respect to longer-term biogeochemical cycling, the model currently lacks any explicit representation of shallow water carbonate deposition in banks and reefs. In terms of global change and anthropogenic impacts on carbon cycling and climate, GENIE is also unable to predict fossil CO2 emissions from human activities (just how much do you want to burn in the future?). Provision is therefore made for prescribing flux fields or boundary conditions to the model. These are termed forcings.

There is a serious difficulty in devising a means for implementing a boundary condition to the model because the most interesting and relevant forcings tend to be time-dependent (i.e., the magnitudes vary with time). Worse, the boundary conditions are often spatially heterogeneous (i.e., the magnitude of the forcing is not the same at each and every grid point or cell). Dust fluxes are a good example of spatial heterogeneity in a flux input to the system. To fully define a time-varying 2D flux forcing for say 1000 years of model integration at a 0.01 yr time-step equates to $36 \times 36 \times 1000 \times 100$ data points – clearly beyond simple and efficient data manipulation and processing. One end-member solution would be to define the complete spatial distribution, but leave the fluxes invariant with respect to time. The opposite end-member would by to fully define the time-varying nature of the forcing but distribute the flux uniformly in space. Both approaches have serious drawbacks for many applications. A compromise scheme has therefore been devised to encapsulate data input and output as efficiently and as flexibly as possible.

Firstly, the forcing of atmosphere, ocean, and sediment tracers are treated separately. For the atmosphere (`atm_*`) tracers, atmospheric composition is manipulated from the 2-D surface ocean grid of BIOGEM. This means that no forcing is applied to the atmosphere overlying 'dry' land cells. For the ocean (`ocn_*`) tracers, all the 'wet' cells in of the full 3-D ocean grid of BIOGEM can be forced. For the sediment (`sed_*`) tracers, all the cells comprising the 3-D ocean grid of biogem (rather than the 2D SEDGEM grid) are manipulated[19].

Secondly, two different types of *forcings* of the system are recognized: a flux forcing and a restoring forcing. The first type, the flux forcing, is pretty self-evident - it represents a flux of tracer that is applied to each cell. The restoring forcing is less intuitive – a flux is applied to the grid cells with a value calculated to bring the tracer values closer to the prescribed restoring value. A time-scale (in years) determines the rate at which the tracer values are brought toward the boundary condition. This time-scale value is termed the restoring constant. The smaller the value of the restoring constant the 'harder' the restoring, and the more rapidly the model will be constrained to approach the boundary condition.

Whether a flux or restoring forcing is required together with the details of the forcing, and what tracer(s) is (/are) to be forced, are specified in a set of 3 tracer forcing configuration files; one for each type of tracer:

- `configure_forcings_atm.dat`
- `configure_forcings_ocn.dat`
- `configure_forcings_sed.dat`

Each file consists of rows of settings; one row for each different tracer. The settings are arranged into columns; the description of each column and allowed values are summarized at the bottom of each file. Options include whether a flux or restoring forcing is required, the time-constant of any restoring forcing, etc.

For a tracer forcing top be fully defined, one further set of files are required, taking filenames of the format:

- `biogem_force_*_sig.dat`

and contains two columns of information: the first is a time marker (year) and is paired with a corresponding tracer scalar value in the second column. These files all live happily together in the directory defined by the namelist parameter: (`i,j`). The information in the signal file define the time-varying information. For a model year falling outside of the maximum or minimum time markers specified in the signal file, no forcing will be applied (this in effect allows a restoring forcing to be turned 'on' and then 'off' again later). For model years inbetween specified time points,

[19] No provision is currently made for restoring forcing.

the tracer scalar values are linearly interpolation. It should be noted that `biogem_force_*_sig.dat` file must contain at least one (row) entry.

The units for the flux forcing for all tracers is mol yr-1 for bulk fluxes, and per mil for isotopic composition. For restoring forcings, the units are atmospheres (atm) for atmospheric (gaseous) tracers, and mol kg-1 for ocean (dissolved) tracers. For isotopic tracers, the units are per mil (‰).

There are then two alternative ways of providing the final information needed to fully describe a tracer forcing – the spatial component – by explicitly defining 2- or 3-D distributions (the sole, original methodology), or by assuming a much simpler, point source, or uniform distribution. The latter is rather easier to grasp and implement, but lack the flexibility of the former.

A variety of example tracer *forcing* configurations are provided in (`genie-forcings`).

4.1 Tracer forcing as an explicit 2/3-D distributions

The tracer forcing in the explicit 2/3-D methodology requires two further files, with filenames:

- `biogem_force_*_I.dat`
- `biogem_force_*_II.dat`

These hold information about spatial distributions. The data format of these is either 2-D for the atmosphere or surface ocean only - rows (`j`) and columns (`i`), or 3-D (for the whole ocean) with the successive depth layers (`k`) repeated as `(i,j)` blocks down through the file.

The way in which these three files are employed to define a spatially-explicit time-varying input field is as follows. The first field file (`*_I.dat`) defines the baseline distribution and the second (`*_II.dat`) an alternative distribution. The difference between the two fields defines the spatial component of a time-varying forcing. The tracer scalar value in the signal file modifies the difference between the two spatial fields.

Putting it another way – the magnitude of the forcing that is applied at any location (`i,j`) and at point in time (`t`), $F_{(i,j),(t)}$, is equal to the baseline field, defined in $biogem\_force\_*\_I_{(i,j)}$, plus the time-dependent modifier ($S_{(t)}$) times the difference between the two fields, i.e.:

$$F_{(i,j),(t)} = biogem\_force\_*\_I_{(i,j)} + S_(t) \times \left(biogem\_force\_*\_II_{(i,j)} - biogem\_force\_*\_I_{(i,j)}\right)$$

Remember that the value of the time-dependent tracer scalar (`S(t)`) is interpolated from the contents of the forcing signal file `biogem_force_*_sig.dat`.

Because what I have just written above is probably vanishingly indistinguishable from complete gobbledygook, a couple of examples might help (or not):

1. One way of using the forcing functionality of BIOGEM would be to assign a minimum forcing field to `I`, a maximum forcing field to `II`, and specify a normalized (i.e., taking values between 0.0 and 1.0) series of modifier values in the the signal file. For example, one could continually vary the surface temperature field seen by the marine carbon cycle over the deglacial transition, based on available end-member reconstructions for glacial maximum and modern [*CLIMAP*, 1980]) and by assuming a semi-representative (normalized) signal with which to interpolate between these two time-slice reconstructions (see *Ridgwell* [2001]).
2. A second way of using the functionality of tracer forcing in BIOGEM would be to set every wet cell in `biogem_force_*_I.dat` to a value of 0.0, and every corresponding location in `biogem_force_*_II.dat` to a value of 1.0. The forcing applied to the model will then be the same as applying the value of the interpolated tracer scalar (in the signal file) equally to each and every cell, i.e.:

   $F_{(i,j),(t)} = 0 + S_(t) \times \left(biogem\_force\_*\_II_{(i,j)} - 0\right)$

   An example of this usage would be in applying a spatially uniform time-varying change in atmospheric composition, such as restoring atmospheric CO2 to an observed historical or predicted future atmospheric concentrations trajectory.

It should be noted that although this scheme is generic and can equally be applied to any tracer (`atm`, `ocn`, or `sed` and including isotopic properties) as well as offering reasonable flexibility in representing the time-varying characteristics of a boundary condition, it does have limitations and cannot cover all possible eventualities. For instance, historical changes in atmospheric CFC concentrations and CO2 radiocarbon activity vary not only with

time, but the spatial heterogeneity changes in a complex way that cannot be represented as a interpolation between two alternative end-member distributions. In this way the built-in framework for tracer *forcings* in GENIE does not conform to the OCMIP protocol [REF] for ocean (carbon cycle) model evaluation.

4.2 Tracer forcing as a point source or uniform distribution

While the spatial forcing fields (`biogem_force_*_I.dat`, `biogem_force_*_I.dat`) provide maximum flexibility, they are not always so easy to use, nor perhaps, understand(!) ;) An alternative, simply way of specifying forcings has therefore been implemented. You can access this by adding the namelist:

```
bg_ctrl_force_oldformat=.false.
```

which simply tells BIOGEM to expect a new file format for the tracer forcing configuration files.

As before, the tracer forcing configuration files live in the directory specified by the namelist parameter `bg_par_forcings_name` e.g.:

```
bg_par_forcings_name="worbe2_historical"
```

There are 3 files - one for each type of tracer (i.e., gas, dissolved, solid/particulate):

- `configure_forcings_atm.dat` – gases
- `configure_forcings_ocn.dat` – dissolved substances
- `configure_forcings_sed.dat` – solid substances / particulates

For instance, the format of the tracer forcing configuration file `configure_forcings_atm.dat` looks like:

```
 01 02   03 04 05 06  07  08
 \/ \/   \/ \/ \/ \/  \/  \/

-START-OF-DATA-
 F  0.0  F  F  F   2  01  01  '[surface air temperature (K)]'
 F  0.0  F  F  F   2  01  01  '[specific humidity (n/a)]'
 f  0.1  f  t  F   2  01  01  '[carbon dioxide (CO2) partial pressure (atm)]'
 f  0.1  f  t  F   2  01  01  '[d13C CO2 (o/oo)]'
 F  1.0  F  F  f   2  01  01  '[oxygen (O2) partial pressure (atm)]'
 F  1.0  F  F  F   2  01  01  '[d18O O2 (o/oo)]'
 F  1.0  F  F  F   2  01  01  '[nitrogen (N2) partial pressure (atm)]'
 F  1.0  F  F  F   2  01  01  '[d15N N2 (o/oo)]'
 F  1.0  F  F  F   2  01  01  '[methane (CH4) partial pressure (atm)]'
 F  1.0  F  F  F   2  01  01  '[d13C CH4 (o/oo)]'
 F  1.0  F  F  F   2  01  01  '[d14C CH4 (o/oo)]'
 F  1.0  F  F  F   2  01  01  '[sulphur hexafloride (SF6) partial pressure (atm)]'
 F  1.0  F  F  F   2  01  01  '[nitrous oxide (N2O) partial pressure (atm)]'
 F  1.0  F  F  F   2  01  01  '[d15N N2O (o/oo)]'
 F  1.0  F  F  F   2  01  01  '[hydrogen sulphide (H2S) partial pressure (atm)]'
 F  1.0  F  F  F   2  01  01  '[d32S of H2S (o/oo)]'
 f  0.1  F  F  F   2  01  01  '[CFC-11 partial pressure (atm)]'
 f  0.1  F  F  F   2  01  01  '[CFC-12 partial pressure (atm)]'
-END-OF-DATA-

 /\ /\ /\   /\ /\ /\  /\  /\  /\
 01 02 03   04 05 06  07  08  09
```

The description of each column is detailed at the bottom of the file:

```
COLUMN #01: include restoring forcing of tracer? [DATA FORMAT: 'T'/'F' ('t'/'f')]
COLUMN #02: time constant of restoring forcing (years) [DATA FORMAT: real]
COLUMN #03: include flux forcing of tracer? [DATA FORMAT: 'T'/'F' ('t'/'f')]
```

```
COLUMN #04: scale flux forcing of tracer? [DATA FORMAT: 'T'/'F' ('t'/'f')]
COLUMN #05: force ocean surface in equilibrium with atmosphere?
            (needs ocean restoring forcing set) [DATA FORMAT: 'T'/'F' ('t'/'f')]
COLUMN #06: make forcing uniform over this dimension
            (2 = 2D, 0 = point, ELSE spatially explicit forcing) [DATA FORMAT: integer]
COLUMN #07: i grid location of point forcing [DATA FORMAT: integer]
COLUMN #08: j grid location of point forcing [DATA FORMAT: integer]
COLUMN END: TRACER DESCRIPTION
```

The columns pertinent to the this alternative way of implementing tracer forcings are:

- `#01` - do you want to restore atmospheric composition toward a specified concentration (or isotopic ratio)? '`t`' for true ('yes'); '`f`' for false ('no'). Note that it does not matter whether you use upper-case (capitals) ('`T`') or lower-case ('`t`'). If selected, column `#2` determines the time-constant of the restoring.
- `#03` - (if not `#1`) do you want to apply a flux of gas to the atmosphere?
- `#06` - what degree of homogenization of the flux do you wish to apply? '`2`' == apply flux evenly throughout the atmosphere; '`0`' apply the flux at a single point only; ('`-1`' == use the original (more complex) flux forcing methodology with the flux or fractional weighting of total flux specified explicitly at each individual grid point). If 0, a forcing-at-a-point, the next 2 columns (`#7` and `#8`) then define the (`i,j`) location on the model grid (longitude ('`i`') and latitude ('`j`')) of the point source.

The forcing configuration files for dissolved tracers (`configure_forcings_ocn.dat`) and particulates (`configure_forcings_sed.dat`) should be self-explanatory; again there is a description of each column at the bottom of the file. However, for dissolved tracer forcings, there is an additional column for the '`k`' (depth level in the ocean) location of a flux applied at a single point, in addition to longitude ('`i`') and latitude ('`j`').

5 Saving data

This section covers *c*GENIE saves data and how to ensure that the variables you want are saved and when you want.

1. Overview.
2. *Time-series* output.
   - *Time-series* file naming conventions.
   - Specifying frequency and timing of *time-series* data saving.
   - Seasonal/monthly data saving.
3. *Time-slice* output.
   - *Time-slice* file naming conventions.
   - Specifying frequency and timing of *time-slice* data saving.
4. Specifying which data fields to be saved in the *time-series* and *time-slice* format.
5. *Re-start* files.

5.1 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 experiment [20].

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.

In the ATCHEM results directory, only the following file will be present:

[20] With the exception of sediment core location environmental properties, which are saved more frequently.

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.

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 biogeochem 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.[21] 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.[22]
4. `biogem_series_*.res`[23] – *Time-series* results file – globally and surface-averaged propoerty 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 includes:
   - time mid-point and integration interval
   - global ocean surface area and volume
   - mean global sir-sea gas exchange coefficient (for CO_2)
   - mean atmospheric tracer concentrations plus total inventory
   - mean ocean tracer concentrations plus total inventory
   - mean plus total global productivity
   - mean plus total global sedimentation

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.[24]
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`[25]. In the mask file, a '`1`' indicates a location to save a sediment core at, and a '`0`' indicates that no sdiment 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.
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.

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.

[21] The mid-points at which time-slices are saved are specified as described above.
[22] The mid-points at which time-slices are saved are specified as described above.
[23] `.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"`
[24] This data is saved only at the termination of an experiment (i.e., the netCDF file contains only a single time-slice).
[25] The 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`.

2. `biogem_series_*` – Time-series results files.

5.2 '*Time-slice*' output

5.2.1 'Specifying 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`[26]. 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 pain text (ASCII) file in the `cgenie.muffin/genie-biogem/data/input` directory, whose name is specified by the parameter `bg_par_infile_slice_name`[27]. For example, the default *time-slice* specification file `save_timeslice.dat` contains the specification[28]:

```
-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 tiem 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 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).

[26] An 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.

[27] The 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`.

[28] The 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.

5.2.2 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*.

5.2.3 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`)[29] 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 ;) *c*GENIE default working employs 96 time-steps per year for a 16-level ocean circulation model (GOLDSTEIN) and 48 for BIOGEM[30]. 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 *c*GENIE, GOLDSTEIN may be operating on 48 time-steps per year, and BIOGEM on 24 or even 12. As cGENIE 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.

5.2.4 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 saving[31]. This can be done by setting:

  `bg_ctrl_data_save_3d_sig=.true.`
  (by default, `.false.`).

[29] Default value = 999

[30] Note 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.

[31] This is in addition to normal 3D saving at the time-slice data saving frequency

5.3 '*Time-series*' output

5.3.1 Specifying 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.[32] 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.[33]

5.3.2 Saving orbital insolation

For evaluating and checking orbital experiments in cGENIE, it is useful to be able to extract commonly used diagnostics of orbital variations in insolation, e.g. June 21 65N. cGENIE 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 18x18 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 18x18x8 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 thsi 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).

5.4 Data field selection

Model output – both *time-slice* and *time-series* data are saved in blocks or categories. For instance, all dissolved tracers in the ocean (3D netCDF *time-slice* and/or *time-series*), particle flux fields, carbonate chemistry, surface sediment composition, etc etc. This requires a multitude of parameters, one for each category and generally also one

[32] For 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.

[33] Note that if running a multi-stage experiment, for the *biogem*, *sedgem*, *rokgem* and *ents* output not to be overwritten each time the model is restarted, you need to set in the *user-config* file:
`bg_opt_append_data=.TRUE.` and `ents_opt_append_data=.TRUE.`
(`sg_ctrl_append_data=.TRUE.` and `rg_opt_append_data=.TRUE.` are set by default)

for each of *time-slice* and *time-series* data. In an attempt to simplify this, a single parameter, `par_data_save_level`, specifying the 'save level' is set instead. The save level 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.
- `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. [MOST COMMON OPTION]
- `5` – Basic + biology + geochemistry diagnostics. Geochemistry diagnostics includes: Fe speciation and fluxes, and water volume production and consumption/oxidation rates. In conjunction with the ROKGEM module, also: weathering fluxes.
- `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 user-specified settings for individual save categories. Default is broadly consistent with previous version of the model.

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*.

5.5 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 *c*GENIE (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.

6 Visualizing *c*GENIE model output

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 *BIOGEM* and *SEDGEM* separately and both modules have a flag that suppresses data file saving in ASCII format, with netCDF format being the default.

Under unix/linux, the contents of a netCDF file 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 section covers how to visualize *c*GENIE (netCDF) spatial data:

1. **Panoply**
   If you really really must insist on using Windozzzz, a 'recommended' viewer for netCDF is Panoply (see below). In fact, the can be run under linux and on the Mac (OS X) as well.
   There is also ncBrowse. Again, this will also run under LINUX and on the Mac (OS X).
2. **MUTLAB**
   You can also process and plot netCDF fields using MUTLAB, for which a number of plotting functions are provided as part of the *c*GENIE code distribution. An advantage here is that the MUTLAB code can be hacked to produce much more powerful and bespoke analysis and plots as well as there being provision for overlying data and calculating model-data statistics as part of the plotting.

6.1 A brief guide to plotting with 'Panoply'

WARNING These instructions are valid only for Panoply version 2.9.4 and the plotting control buttons and options may have subtly changed in newer versions ...

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 3-D mode output, you will be asked first whether you want a 'Lon-Lat' or 'Lat-Vert' plot, and in newer releases of Panoply, 'Lon-Vert'. (. For the 2-D fields there is no choice and the plot display will immediately open.

- For 'Lon-Lat' plots - there are multiple levels (depth layers) in the ocean of data that can be plotted, from the surface to the abyssal ocean.
- For 'Lat-Vert' plots - there are multiple possible longitudes at which to plot slices. The default is of the global mean meridional distribution.
- For 'Lon-Vert' plots - as 'Lat-Vert' but latitude against depth in the ocean.
- For all slices: there may be multiple time-slices (i.e., you can plot data saved from different years) as well as longitude/latitude/depth.

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. 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 ...` then select the location, filename, and graphics format. It is also possible to create and save animations.

6.1.1 Issues with Panoply default settings

NOTE: The default settings and the way in which Panoply reads in the data array can sometimes mislead. In particular:

1. **Year** (**Array** tab)
   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 '**equal month years**').
   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.
2. **Scale Range** (**Scale** tab)
   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).
3. 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.
4. 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 **-6** in effect converts units of mol kg^{-6} to μmol kg^{-6}.
5. Array position counting.
   Panoply assigns a integer count to the position in the array as is was written. For time, this makes some sense – the first saved *time-slice* is reported as '`year 1`' [of n]. The second saved *time-slice*, '`year 2`' etc. Not so usefully for depth in the ocean, the surface layer (displayed by default in a 'Lon-Lat' plot becomes 'Z level mid depth **1**' [of n], whereas, for a 16-level ocean, the *c*GENIE grid counts this as **16** ...

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).

6.1.2 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 variability in that variability. To do this:

1. First, open the netCDF results file.
2. Open the variable of interest, e.g., `atm_temp` (surface air temperature) in the 2D netCDF file.
3. 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** ...).
4. From the upper LH corner of the Dataset Browser window, now click on the Combine Plot icon.
   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).

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. Also note that a general scientific convention is to have a centered scale so that no change is white, with positive deviations = red and negative = blue. The same variable in two different model experiments can also be opened up ain an analagous manner.

As an (simpler?) alternative – it is possible to open up one plot, and then drag a variable (either the same variable from the same experiment, a different variable from the same experiment, or a variable form a different experiment) into the open plot window. A difference (combined) plot is automatically generated.

6.1.3 Velocity (ocean current direction and magnitude) 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. Taking the simpler drag-and-drop approach ...

1. First, open the 3-D netCDF results file.
2. Open either the `phys_u` ('ocean velocity - u') or `phys_v` ('ocean velocity - v') field and select a Lon-Lat plot.
3. From the Dataset Browser window, drag the other valocity variable into the open plot window.
4. 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 2. To display deeper (in the ocean) current fields and/or different time-slices, 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.

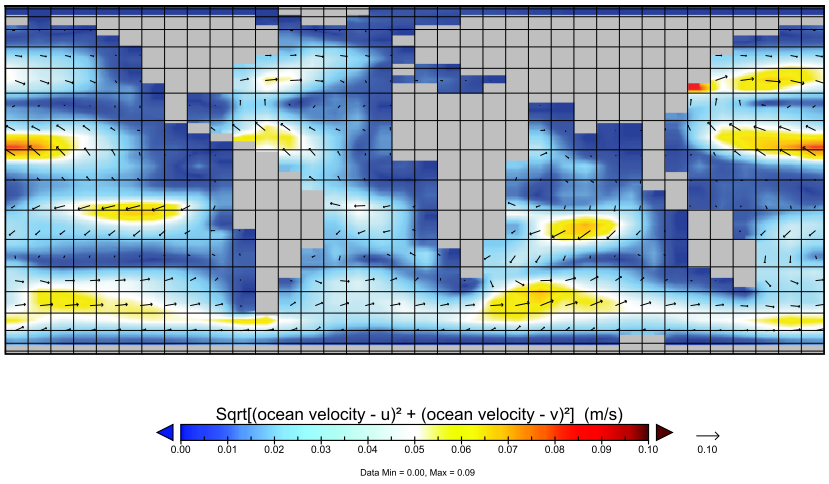

Figure 2: Example (modern) ocean surface velocity (current) map.

6.2 MUTLAB plotting – 2D and 3D fields

Functions are provided for plotting BIOGEM and SEDGEM netCDF output. These can be found in the `cgenie.muffin` sub-directory: `genie-matlab`, and are:

1. `plot_fields_biogem_2d.m` – Plot a 2-D field from: `fields_biogem_2d.nc`.

2. `plot_fields_biogem_3d_i.m` – 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 also 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 provided to overlay an ocean circulation stream-function.
3. `plot_fields_biogem_3d_k.m` – Plot a horizontal slice through the ocean. An option is provided for overlaying ocean circulation velocity vectors. Water column integrals can also be calculated and displayed.
4. `plot_fields_sedgem_2d.m` – 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.

A little configuration is required before they can be used.

1. Firstly, pick a directory that will be your working directory where all the graphical output will be stored and data sets read in from and written out to. The MUTLAB working directory will need to be set to this directory. Asusming your working directory is `C:\cgenie.analysis`, you'll need to change to this directory at the command line in MUTLAB:
   `>> cd('C:\cgenie.analysis');`
2. Secondly, MUTLAB has to know where the plotting functions and their ancillary files live. Simplest, but messiest, is to copy the contents of `genie-matlab` to your working directory (the directory where you intend to generate the output).
   The contents of `genie-matlab` can live elsewhere, but the path to this must be added to MUTLAB's search path, by:
   `>> addpath('C:\cgenie.resources\matlab');`
   for an example of all the files living in the directory `cgenie.resources\matlab` on the `C` drive. The path can also be set in `plot_fields_settings.m` – see below.
3. MUTLAB also needs to know where to find the cGENIE experiment results. Simplest, and consistent with the default settings, is to create a directory `cgenie_output` in your working directory, and copy the experiment results there. i.e. if the working directory was `C:\cgenie.analysis`, you would create a directory `C:\cgenie.analysis\cgenie_output` and copy the experiments to be processed there (e.g., so that there might be an experiment: `C:\cgenie.analysis\cgenie_output\EXAMPLE.worjh2.Caoetal2009.SPIN`). Alternatively, the path to the results can be set in `plot_fields_settings.m` – see below.

One the MUTLAB environment is correctly configured, 'help' can be obtained on the plotting functions by e.g. typing:

```
>> help functionname
```

where `functionname.m` is one of the '`.m`' files (listed above).

6.2.1 Argument (parameter) list

All 4 plotting functions share exactly the same format of **comma-separated** parameters[34] passed in the argument list of the plotting function in the form: `functionname(PAR1,PAR2,PAR3, ... PARn)` where n is 15(!) ...

The first section of passed parameters defines the experiment to be plotted, as well as variable year, and if multiple experiments/years/variables are given, that a difference (anomaly) map is to be generated. They are:

- `PEXP1` – is the name of the 1st (main) experiment.[35]
- `PEXP2` – is the name of the 2nd (optional) experiment. If no second experiment is selected, then a null string value must be passed, i.e., `''`.

[34] Parameters 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`.

[35] As a string, the value must be encased in inverted commas: `''`.

- PVAR1 – 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 and a)valid) variable name can be re-entered.
- PVAR2 – is the name of the 2nd (optional) variable. If no second variable is selected, then a null string value must be passed, i.e., ''.
- PT1 – is the value of the 1st (main) time-slice. If no valid variable value is given, a list of valid times (years) will be printed out. As *SEDGEM* does not save multiple and/or time-specific data in the fields_sedgem_2d.nc results file, a dummy value (any real number) is entered here.
- PT2 – is the value of the 2nd (optional) time-slice. If no second time-slice is selected, enter a value of -1.

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):

- PIK – varies in its interpretation and is discussed below.
- PMASK – is the name of an optional (2D) mask. A null string ('') must be passed if no mask is requested. The interpretation of this parameter differs slightly between functions (below).

Next come options for plotting scale control:

- PCSCALE – 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 and the minimum and maximum limits (below) ignored if a value of zero (0.0) is entered.
- PCMIN – is the minimum scale value.
- PCMAX – is the maximum scale value.
- PCN – is the number of (contour) intervals between minimum and maximum scale values.

Finally, there is a parameter for specifying discrete (observed) data to be plotted:

- PDATA – is the filename containing the an overlay data set. The full filename of this file must be give, including any extensions. This parameter must be passed as a string; leave blank, i.e., '', for no overlay data.
  The format of the file depends on the plotting function:
  - plot_fields_biogem_2d.m and plot_fields_sedgem_2d.m The data must be formatted as 4 separated columns of: lon, lat, value, label.
  - plot_fields_biogem_3d_i.m and plot_fields_biogem_3d_k.m The data must be formatted as 5 separated columns of: lon, lat, depth, value, label.

  Dummy values can be given for e.g. depth. For labels, dummy values are OK, as is a blank space. But note that a label string must contain no spaces (i.e. it must be a continuous string of characters).

then:

- PDOPT – is a string that points to a filename containing a specific plotting parameter set (see later). By default (i.e. '' is entered), the file used is plot_fields_settings.m. Enter the name of an alternative filename, **omitting** the .m extension, to substitute the default parameter set. E.g. 'anomaly_plotting_settings' would result in the m-file anomaly_plotting_settings.m to be automatically loaded when the plotting function is run. The content of this parameter file is described below.

and lastly:

- PFILE – is a string that enables a specific (root) filename to be prescribed. Leaving this parameter blank (e.g. '') results in output filenames being automatically generated.

The basic parameter list to all 4 plotting functions is hence:

```
functionname(PEXP1,PEXP2,PVAR1,PVAR2,PT1,PT2,PIK,PMASK,PCSCALE,PCMIN,PCMAX,PCN,PDATA,POPT,PNAME);
```

For example:

```
plot_fields_sedgem_2d('run1','','sed_CaCO3','',0.0,0.0,0,'',1.0,0.0,100.0,20,'','','')
```

plots plot the carbonate content of surface sediments from experiment 'run1', between 0 and 100 wt% in 20 contour intervals. The default set of plotting control parameters are used (by means of the last ''). The section ',0.0,0.0,0,'','' near the middle are the dummy variables (time-slice and alternative time-slice, i/k value, and mask) introduced for consistency in format with the other 3 plotting functions.

And
`plot_fields_sedgem_2d('run1','','sed_CaCO3','',0.0,0.0,0,'',1.0,0.0,100.0,20,'caco3.dat','','')`
is much the same except now with an overlain observational dataset defined in the file `caco3.dat`.

Note that for `plot_fields_sedgem_2d.m` several of the parameters are redundant but **must** be included (typically as zeros). This is in order to retain a common parameter list format between all the different plotting functions.

6.2.2 Function specific interpretation of PIK and PMASK

`PIK` and to some extent `PMASK` have quite different interpretations depending on which plotting function is being used:

1. `plot_fields_biogem_2d.m`
   - `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 of equal to the passed value will be plotted.
   - `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.m`
   - `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` (e.g. *Cao et al.* [2009]). (`plot_fields_biogem_3d_i.m`) to be plotted. For `fields_biogem_2d.nc`, Zero values have special meanings here – in `plot_fields_biogem_3d_k.m`, a zero will result in a water column inventory map being plotted.
   - `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.m`
   - `PIK` – the longitude-depth (`i`) slice through the ocean to be plotted.
   - `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' longtudinally, but which take a meandering path following an ocean transect[36].
     The trivial usage would be to construct a mask consisting of a vertical line of `1`s – 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 `MASK` is set) 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 masks can be found in the `genie-matlab` directory.

6.2.3 Further plotting control and refinements

A large number of additional settings are provided for exerting finer control over the plotting. These additional setting are provided in the form of a block of parameters with associated (default) values in a separate m-file – `plot_fields_settings.m`. As an m-file, `help plot_fields_settings.m` will call up help on the settings with a brief description. Again – note that the parameter file is common to all 4 plotting functions and as such not all the options are relevant to all of the plotting functions[37].

[36] e.g., as in: `mask_worjh2_GEOSECS_WATL.dat`
[37] See 'help' on a specific plotting function for details of the relevant options in the parameter block.

The default file (plot_fields_settings.m) can be edited (and saved) in order to change the plot. Or (better), the default parameter file can be copied and renamed and commonly used combinations of settings saved in a series of alternative files[38]. The file used is determined by the string value of the 14th (last) passed parameter – POPT (see above). Remembering to **omit** the .m extension when specifying the filename ...

The full list (with defaults in brackets []) is as follows:

CONFIGURATION PATHS

- data_path='cgenie_output'; ['cgenie_output'] Is the relative pathway to the directory location of the experiments. For example, if all the individually-named experiment directories are located in a directory named cgenie_output (as per how results are saved by default by the model), and this directory is located in the same directory as the MATLAB function is being run, parameter PATH takes the value: 'cgenie_output'.[39] If the experiment directory was located in the same directory as the MATLAB function, a null value must be set, i.e., ''.
- library_path1 = 'C:\cgenie.muffin\genie-matlab'; Is the absolute path to the directory containing the supporting plotting function files (i.e. the contents of the genie-matlab directory).

OVERLAY CONTOUR CONTROL

- contour_plot = 'n'; ['n'] OVERLAY CONTOL PLOT?
  Overlay line contours on the color block plot?
- contour_mod = 2; [2] NUMBER OF COLOR INTERVALS PER CONTOR
  Number of color graduations per line contour.
- contour_mod_label = 4; [4] NUMBER OF LABELED CONTOURS PER CONTOUR
  Number of color graduations per labeled line contour.
- contour_label = 'y'; ['y'] LABEL CONTOURS?
  Label the line contours (frequency of labeled contours set by contour_label.
- 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.)
- contour_dashneg = 'n'; ['n'] PLOT NEGATIVE CONTOURS DASHED?
  Plot negative contours as dashed lines?
- contour_hlt = 'n'; ['n'] ADD HIGHLIGHT CONTOUR?
  Add a bold highlight contour?
- contour_hltval = 0.0; [0.0] HIGHLIGHT CONTOUR VALUE
  Value of the highlight contour (if selected).
- contour_zero = 'y'; ['y'] PLOT ZERO CONTOUR
  Highlight the zero contour?
- contour_file = ''; [''] OPTIONAL EXTERNAL PLOTTING SCALE
  Filename of optional external plotting scale[40].

COLOR AND SCALING

- colorbar_inv = 'n'; ['n'] INVERT COLORBAR
  Invert colorbar colors?
- data_log10 = 'n'; ['n'] PLOT LOG10 OF THE DATA
  Plot log10 of data?
- data_offset = 0.0; [0.0] APPLIED DATA OFFSET
  Introduce a data offset? This is useful for example for converting K to degrees C (removing the K value (273.15) of 0.0 degrees C).

DATA FORMAT

- data_ijk = 'n'; ['n'] DATA AS (i,j,(k)) VS. (lon,lat,depth)?
  Overlay data in the form of (i,j,k) locations rather than longitude, latitude, depth?

[38] In copying the default file, it is not necessary to also copy the block of commented 'help' test at the start of the file and the format can be rather more compact than in the default file.
[39] Note that as a string value, the inverted commas are important.
[40] Currently only implemented in plot_fields_biogem_3d_i.m

- `data_ijk_mean = 'n'; [ 'n'] AVERAGE DATA BY MODEL CELL?`
  Average overlay data per *c*GENIE grid cell rather than plotting raw locations.
- `data_shapecol = 'n'; [ 'n'] DATA COLUMNS TO SET POINT SHAPE AND COLOR?`
  Assume additional columns for point shape and color are present in the data file?

DATA PLOTTING

- `plot_land = 'n'; [ 'n'] PLOT DATA OVER LAND?`
  Plot data locations lying over land on the *c*GENIE grid (rather than screen out)?
- `data_anomoly = 'n'; [ 'n'] PLOT AS MODEL-DATA ANOMOLY ONLY?`
  Plot data locations with the model-data anomaly rather than data value?
- `data_only = 'n'; [ 'n'] PLOT ONLY DATA (no model values)?`
  Plot only the overlay data locations (and not any model data)?
- `data_siteonly = 'n'; [ 'n'] PLOT DATA AS SITES (no data values)?`
  Plot labeled site locations (no data value fill).
- `data_size = 25.0; [25.0] SIZE OF OVERLAY DATA POINTS`
  Size of the overlay data points.
- `data_sitelabel = 'n'; [ 'n'] LABEL SITES?`
  Label data locations?
- `data_sitecolor = 'k'; [ 'k'] SITE MARKER COLOR`
  Color symbol of the site marker.
- `data_sitelineth = 1.0; [ 1.0] SITE MARKER LINE THICKNESS`
  Line thickness of the site marker (pt).
- `data_fontsz = 8; [ 8] LABEL FONT SIZE`
  Font size of data label.
- `data_stats = 'n'; [ 'n'] CALCULATE AND PLOT STATS?`
  Calculate and plot model-data statistics?
- `data_fit_n = 1; [ 1] POLYNOMIAL FIT ORDER`
  Polynomial fit order for cross-plotting (model vs. model, or data vs. model).

REPLACEMENT ASCII-FORMAT DATA FIELDS

- `plot_dataid_alt1=''; [ ''] DATA FIELD no.1 REPLACEMENT FILE`
  Datafile (ASCII) replacement for 1st spatial field.
- `plot_dataid_alt2=''; [ ''] DATA FIELD no.2 REPLACEMENT FILE`
  Datafile (ASCII) replacement for 2nd spatial field.

GRID PLOTTING

- `plot_landvalues='n'; [ 'n'] PLOT VALUES OVER LAND?`
  Plot values over land (for 2D data)?
- `plot_mask_netcdf = 'grid_mask_dsea'; [ ''] INTERNAL netCDF MASK NAME`
  Set internal netCDF mask name.

DATA AVERAGING/WATER COLUMN INTEGRAL CONTROL

- `data_av_kmin = 1; [ 1] MIN k FOR AVERAGING`
  The minimum k value used for averaging.
- `data_av_kmax = 1; [ 1] MAX k FOR AVERAGING`
  The maximum k value used for averaging.
- `plot_av_conc = 'n'; [ 'n'] PLOT AVERAGE CONC RATHER THAN INVENTORY?`
  Plot the average as a concentration (the default is for a water column inventory).
- `plot_minval = 'n'; [ 'n'] PLOT MINIMUM WATER COLUMN VALUE?`
  Plot the minimum val in the column interval?
- `plot_maxval = 'n'; [ 'n'] PLOT MAXIMUM WATER COLUMN VALUE?`
  Plot the maximum val in the column interval?

CURRENT VELOCITY OVERLAY

- `data_uv = 'n'; [ 'n'] overlay (u,v) velocity data?`
  Overlay ocean current fields. Not valid for `plot_fields_biogem_3d_i.m` or `plot_fields_sedgem_2d.m`.

- `data_uv_scale = 1.0; [ 1.0] scaling factor for vector length`
  Scaling factor for velocity vectors.

`STREAMFUNCTION OVERLAY`

- `plot_opsi = ''; [ ''] PLOT OVERTURNING STREAMFUNCTION (basin)?`
  Overlay stream-function (''==NONE): 'g'==global; 'a'==Altantic; 'p'==Pacific. NOTE: only valid for `plot_fields_biogem_3d_i.m`.
- `plot_opsi_min = -20; [ -20] MINIMUM STREAMFUNCTION VALUE`
  Stream-function minimuim (Sv).
- `plot_opsi_max = +20; [ +20] MAXIMUM STREAMFUNCTION VALUE`
  Stream-function maximum (Sv).
- `plot_opsi_dminor = 2; [ 2] MINOR STREAMFUNCTION CONTOUR INCREMENT`
  Stream-function minor contour interval (Sv).
- `plot_opsi_dmajor = 4; [ 4] MAJOR STREAMFUNCTION CONTOUR INCREMENT`
  Stream-function label interval (Sv).

`PLOT FORMAT AND DIMENSIONS`

- `plot_main = 'y'; [ ''] PLOT THE MAIN FIGURE`
  Plot the main figure?
- `plot_secondary = 'y'; [ ''] PLOT SECONDARY FIGURES`
  Plot secondary figures?
- `plot_format_old = 'y'; [ 'y'] 'OLD' STYLE PLOTTING`
  Choose 'old' style plotting? note that 'new' plotting ('y') requires additional Windows library resources.
- `plot_format = ''; [ ''] FORMAT OF (NEW STYLE) PLOT`
  Format of 'new' syle plot (if above is 'y'): options: 'jpg', 'png', 'eps', 'pngT' adds transparency to png.
- `plot_equallat = 'n'; [ 'n'] PLOT WITH EQUAL LAT INCREMENTS`
  Plot with equal lat increments (rather than as equal area).
- `plot_lon_min = -180; [-180] STARTING LONGITUDE FOR X-AXIS`
  Sets the longitude of the left-hand edge of the plot.
- `plot_lon_delta = 90; [ 90] INCREMENT OF LONGITUDE ON X-AXIS`
  Sets the longitude tick increment.
- `plot_title = ''; [ ''] OPTIONAL REPLACEMENT TITLE`
  String for the alternative plot title. This parameter must be passed as a string, e.g., `'distribution of bottom-water phosphate concentrations'`. If an empty (i.e., `''`) value is passed to this parameter then a title is automatically generated.
- `plot_dscrsz = 0.60; [0.60] FRACTIONAL FIGURE WINDOW SIZE`
  Adjustment factor of the fractional size (compared to the screen) of the figure window.
- `plot_kmin = 0; [ 0] OPTIONAL MAX PLOTTING LIMIT (IN K-SPACE)`
  Optional k-space plotting limts – min[41].
- `plot_kmax = 0; [ 0] OPTIONAL MAX PLOTTING LIMIT (IN K-SPACE)`
  Optional k-space plotting limts – max[42].

`OPTIONAL ADDITIONAL PATHS`

- `library_path2 = 'C:\cgenie.muffin\genie-matlab\xpdfbin-win-3.03\bin32';`

- `library_path3 = 'C:\cgenie.muffin\genie-matlab\export_fig';`

6.2.4 General plotting examples

Examples of general plotting:

1.

[41] Enacted *only* when plot_kmin is not equal to plot_kmax
[42] Enacted *only* when plot_kmin is not equal to plot_kmax

6.2.5 Data plotting examples

Examples of data plotting:

1. **Data plotting – example #1**
   Various combinations of model and data (overlay) can be plotted depending on the selection of the options within the m-file.

   The options given below are the for example of `plot_fields_sedgem_2d` – the other plotting functions may not have exactly the same options (or even necessarily the exact same nomenclature) depending on the respective state of updating of the MUTLAB code ...

   - `data_ij = 'n'; [ 'n'] DATA as (i,j)?`
     Overlay data in the form of (i,j) locations rather than longitude,latitude?
   - `data_ij_mean = 'n'; [ 'n'] average DATA by cell?`
     Average overlay data per *c*GENIE grid cell rather than plotting raw locations.
   - `data_land = 'n'; [ 'n'] PLOT DATA OVER LAND?`
     Plot data points lying over land.
   - `data_anomoly = 'n'; [ 'n'] PLOT AS MODEL-DATA ANOMOLY ONLY?`
     Plot data locations with the model-data anomaly rather than data value?
   - `data_only = 'n'; [ 'n'] PLOT ONLY DATA (no model values)?`
     Plot only the overlay data locations (and not any model data)?
   - `data_siteonly = 'n'; [ 'n'] PLOT DATA AS SITES (no data values)?`
     Plot labeled site locations (no data value fill).
   - `data_sitelabel = 'n'; [ 'n'] LABEL SITES?`
     Label the locations.
   - `data_fontsz = 12; [ 12] LABEL FONT SIZE`
     Font size (pts) of the data labels.
   - `data_size = 25.0; [25.0] SIZE OF OVERLAY DATA POINTS`
     Size of the overlay data points.

   The example use a late Cretaceous simulation of the distribution of CaCO3 in deep-sea sediments, compared to a compilation of late Maastrictian observations (but there is nothing special about specific model experiment and dataset chosen). All the plots are created with the same command line:

   ```
   plot_fields_sedgem_2d('cgenie_output','130510.p0067f.2000alk_0p2RR_detL0.Archer.SPIN1','',
   'sed_CaCO3','',0.0,0.0,0,'',1.0,0.0,100.0,20,'wt% CaCO3','KPgwtpct.dat')
   ```

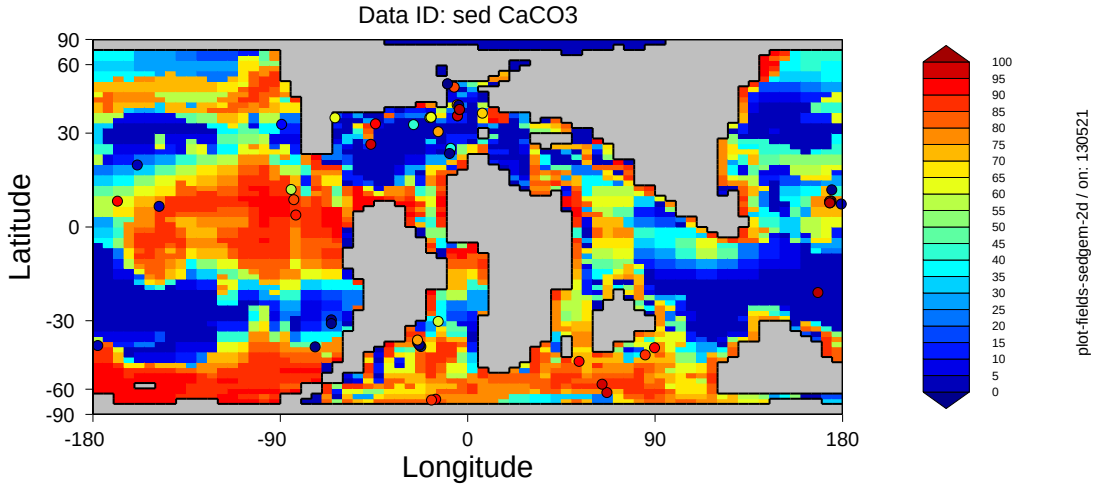

Figure 3: Example *1a*: default [default settings].

Firstly – the data format *must* be 4 columns (although annoyingly, other m-files might differ currently ...). The format is: **lon**, **lat**, **value**, **label** arranged in space-separated columns. If there is no label (or any labels at all) – simply include a dummy text string (e.g. '–') in the 4th column. Text strings must be continuous with no spaces, so adding e.g. an underscore ('_') is needed if the label is 2 separate words. **lon** and **lat** are in degrees

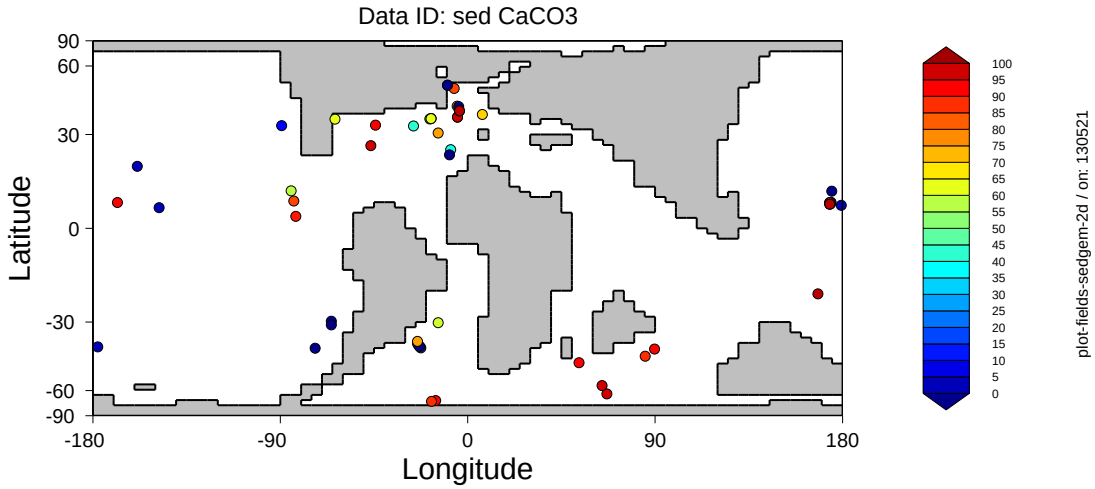

Figure 4: Example *1b*: data only [`data_only = 'y'`].

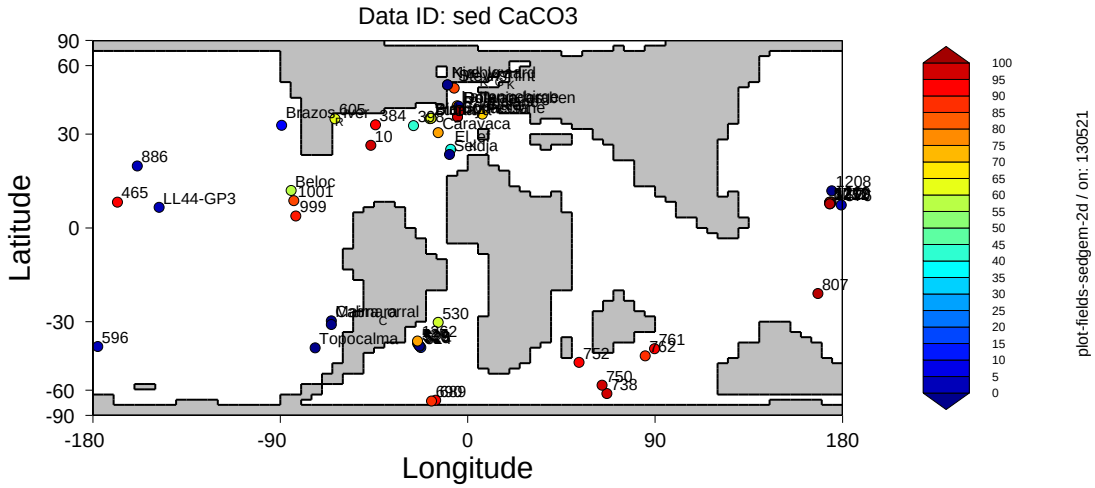

Figure 5: Example *1c*: data only, with site labels and smaller font size [`data_sitelabel = 'y' data_fontsz = 12`].

E and N, respectively with no degree symbol of 'N' allowed (i.e. as a numeric value only). The reference for the longitude does not matter: i.e. if could run from -180.0 to 180.0, or from 0.0 too 360.0 – the m-file will handle this.

The data can alternatively be given in the form of *c*GENIE grid (i,j) grid points, in which case you must set: `data_ij = 'y'`.

There is also the option of whether to re-grid from (lon,lat) and average the data on the cGENIE (i,j) grid. Set: `data_ij_mean = 'y'` for this.

2. **Data plotting – example #2**
   `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`.

6.2.6 Data extraction examples

Examples of extracting model data.

There are occasions where model fields are not needed to be plotted *per se*, but the values at specific locations are wanted. Obviously, one could attempt to find the location and read off the equivalent color from the scale (a *bad idea*). Slightly better is to e.g. use Panoply and view the grid of values (rather than the plotted field); find the location

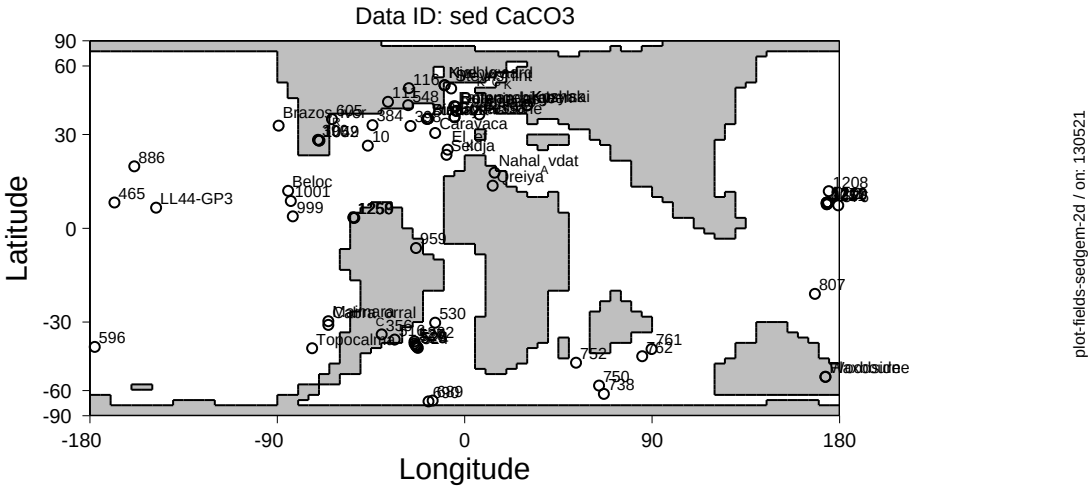

Figure 6: Example *1d*: data locations only, with site labels and also plotting locations over land [`data_sitelabel = 'y' data_only = 'y' data_land = 'y'`].

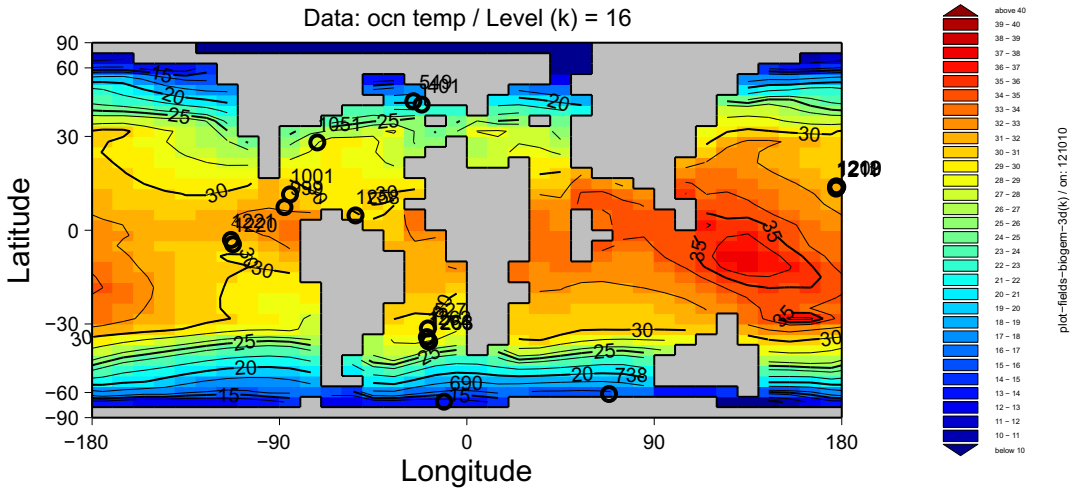

Figure 7: Paleocene-Eocene deep-sea sediment drill locations together with a contour-overlain map of surface temperature.

desired etc – OK for a single point, but rater tedious for many points.

Instead, the MUTLAB plotting functions can be used to determine model field values corresponding to specific locations. There are a number of possibilities, for which examples are given as follows:

1. **Extraction of a single location.**
   This can be done easily in `plot_fields_biogem_3d_k.m`.
   First create a (ASCII) data file (`data.dat` in this example) containing the lon, lat, and depth of the data to be extracted, e.g.

   ```
   %   lon      lat      fake_depth  fake_value  label
       -155.0   25.0     39.0        1           watery_location
   ```

   In this example, a dummy value (`1`) is given for the data value, whilst the label is hardly needed. Pass the data file to `plot_fields_biogem_3d_k.m`. The only complication is that the 'k' level passed must correspond to the depth given in the data file[43]. For example (**all one line**):

   ```
   plot_fields_biogem_3d_k(
   'EXPERIMENT','','ocn_temp','',9999.5,-1,16,'',1.0,0.0,30.0,30,
   'data.dat','','FILENAME');
   ```

[43] A list of cGENIE depth boundaries and midpoints is given earlier on in this manual.

No settings need to be changed compared to the defaults in `plot_fields_settings.m` and so a filename of plotting settings need not be passed. (The optional filename (`FILENAME`) is also ... optional.)
The result is a file called `FILENAME_MODELPOINTS.DATE.dat` and looks like:

```
% Model value at data locations
% Format: i, j, k, lon, lat, depth, model value, data value, label
11 26 16 -155.000   25.000   39.000 2.127626e+001 1.000000e+000 watery_location
```

Both model and data values are given, plus the equivalent i,j,k values to the lon,lat,depth. If the i,j,k location is known *a priori*, the data file can have i,j,k values in place of lon,lat,depth. In this case, a change to the default plotting settings is needed:

```
data_ijk = 'y';         % [ 'n']  DATA AS (i,j,(k)) VS. (lon,lat,depth)?
```

2. **Extraction of a water column profile #1.**
Multiple depths at the same grid point, or more normally – an entire water column profile for a specific lon-lat location, are best extracted using `plot_fields_biogem_3d_i.m`.
The first way is to construct a (ASCII) data file explicitly containing the lon, lat, and depth[44] of the data to be extracted, e.g.

```
%   lon     lat     fake_depth  fake_value  label
    -155.0  25.0    39.0        1           watery_location
    -155.0  25.0    126.0       1           watery_location
    -155.0  25.0    227.0       1           watery_location
    -155.0  25.0    345.0       1           watery_location
    -155.0  25.0    481.0       1           watery_location
    -155.0  25.0    640.0       1           watery_location
    -155.0  25.0    824.0       1           watery_location
```

Pass the data file to `plot_fields_biogem_3d_i.m`, e.g.:

```
plot_fields_biogem_3d_i(
'EXPERIMENT','','ocn_temp','',9999.5,-1,0,'',1.0,0.0,30.0,30,
'data.dat','','FILENAME');
```

Again – settings need to be changed compared to the defaults in `plot_fields_settings.m` and so a filename of plotting settings need not be passed. Note that a value of '0' is passed for the `iplot` variable – this tell the plotting function to use all grid points rather than extract a specific section.
The output looks like:

```
% Model value at data locations
% Format: i, j, k, lon, lat, depth, value, label
11 26 16 -155.000  -25.0  -39.000 2.267929e+001 watery_location
11 26 15 -155.000  -25.0 -126.000 2.066724e+001 watery_location
11 26 14 -155.000  -25.0 -227.000 1.768933e+001 watery_location
11 26 13 -155.000  -25.0-345.000 1.358642e+001 watery_location
11 26 12 -155.000  -25.0 -481.000 9.536270e+000 watery_location
11 26 11 -155.000  -25.0 -640.000 6.808005e+000 watery_location
11 26 10 -155.000  -25.0 -824.000 5.104891e+000 watery_location
```

3. **Extraction of a water column profile #2.** `plot_fields_biogem_3d_i.m` also has the built-in capability of extracting water column profiles – often used on the basis of a basin or regional mean, but equally applicable to a single grid location. You'll need a mask defining the location to have the water column profile extracted from. For example – copy `mask_worjh2_ALL.dat`, search-and-replace all values of '1' to '0', and then mark with a '1', the particular grid location required. The command[45] is then:

```
plot_fields_biogem_3d_i(
'EXPERIMENT','','ocn_temp','',9999.5,-1,0,
'MASKNAME.dat',1.0,0.0,30.0,30,'','','FILENAME');
```

In addition to the water column profile being plotted, the values are saved as `FILENAME.PROFILE.DATE.res`.

[44] A list of cGENIE depth boundaries and midpoints is given earlier on in this manual.
[45] In this example, is does not matter what value is passed for `iplot`, as the presence of a mask over-writes the setting.

6.3 MUTLAB plotting – synthetic sediment cores

A MUTLAB functions is provided for plotting the SEDGEM sediment core netCDF output: `plot_sedcore.m` which will plot a series of vertical columns, as a function of time or stratigraphic depth, including bulk and isotopic composition of carbonate and basic core stratigraphic information, including the age model, accumulation rate, and the location of the any marker of the experiment start. Up to 2 additional columns can be plotted (any of the sedcore variables). `plot_sedcore.m` can be found in the `cgenie.muffin` sub-directory: `genie-matlab` and has on-line 'help' which can be called up by typing:

```
>> help plot_sedcore
```

The format for calling the `plot_sedcore` plotting function is to pass a series of **comma-separated** parameters[46] passed in the argument list of the plotting function in the form: `plot_sedcore(PEXP,PCORE,PMIN,PMAX,PREFAGE,PDATA1,PDATA2,POPT)`

As per the 'help' description, these parameters are:

- `PEXP` [STRING] – is the the experiment name (e.g. 'preindustrial_spinup').
- `PCORE` [STRING] – is the location id of the sedcore to be plotted (e.g. '0102'). All valid sedcore locations will be listed if an invalid name is given.
- `PMIN` [REAL] – is the minimum plotted height/time (e.g. -50.0), as: (i) an anomoly compared to a reference [opt_relreflevel = true] (ii) absolute relative to the suface [opt_relreflevel = false] with +ve. values representing younger or stratigraphically higher than the reference height/time.
- `PMAX` [REAL] – is the maximum plotted height/time (e.g. -50.0).
- `PREFAGE` [REAL] – is the reference age, usually the run length (e.g. 50.0) (age 0.0 is also a valid value). Enter any value less than 0.0 (e.g. -1) to automatically identify the run start and plot as a function of depth relative to this.
- `PDATA1` [STRING] – is the variable name for additional data to plot (e.g. 'sed_LiCO3_7Li').
- `PDATA2` [STRING] – is the variable name for additional data to plot (e.g. 'sed_CaCO3_44Ca').
- `POPT` [STRING] – is the the string for an alternative plotting parameter set (e.g., 'plotting_config_2'). If an empty (i.e., '') value is passed to this parameter then the default parameter set is used (plot_sedcore_settings).

It is important to recognize the primary functionality enabled by the choice of passed parameter values and the main (top-most listed) settings in `plot_sedcore_settings.m`:

1. A non-negative value for `PREFAGE` together with `opt_relreflevel = true` (the default). Will produce a plot with time on the y-axis, with time relative to the reference age set by the value of `PREFAGE`. Time is positive upwards in the core (i.e. towards the surface) above the reference age, and negative below it. The value of `PMIN` must therefore be negative and `PMAX` positive.
2. A negative value (and negative number will do) for `PREFAGE` together with `opt_relreflevel = true` (the default). Will produce a plot with depth on the y-axis, with height relative to the stratigraphic marker ('ash') marking the start of the experiment. Height is positive upwards in the core above the marker horizon, and negative below it. The value of `PMIN` must therefore be negative and `PMAX` positive. Note that the parameter `par_expduration` needs to be set equal to the experiment duration for the age model to be correctly scaled (i.e. the assumed age of the ash peak needs to be correctly assigned).
3. A non-negative value for `PREFAGE` together with `opt_relreflevel = false`. Will produce a plot with time on the y-axis, with time relative to the surface (zero). The values of both `PMIN` and `PMAX` must be negative or zero (with `PMIN` more negative than `PMAX`). Note that the same can be achieved (more easily) by setting a zero reference age (i.e. `PREFAGE = 0.0`.)
4. A negative value for `PREFAGE` together with `opt_relreflevel = false`. Will produce a plot with depth on the y-axis, with height relative to the surface. The values of both `PMIN` and `PMAX` must be negative or zero (with `PMIN` more negative than `PMAX`).

For example:

```
plot_sedcore('run1','0131',-50.0,50.0,100.0,'','','basic_settings')
```

[46] Parameters 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`.

plots core location `0131` from experiment '`run1`'. A reference horizon at 100 kyr is assumed, with sediment properties plotted as a function of age[47] between -50.0 and +50.0 kyr of this horizon. No additional (other than the basic 5 panels) tracers are requested for plotting. Other plotting parameters are taken form the file: `basic_settings.m`.

Refer to the description of additional parameters in the header of the default settings file: `plot_sedcore_settings.m` for additional help and information on plotting. Some warnings/errors are reported if parameters result in an out-of-range situation.

[47] Age rather than depth is assumed for the y-axis because the passed value of the reference horizon parameter, `PREFAGE` , is non negative.

A Available GENIE modules

Component	Abrv.	?	Model	Reference
Atmosphere	ig eb fa		IGCM (3-D AGCM) EMBM (2-D) Fixed	*de Forster et al.* [2000] *Fanning and Weaver* (1996) —
Ocean	go so fo		GOLDSTEIN Slab Fixed	*Edwards and Marsh* (2005) *de Forster et al.* [2000] —
Sea-ice	gs ss fs		GOLDSTEIN (multi-option) Slab Fixed	*Edwards and Marsh* (2005) *de Forster et al.* [2000] —
Land surface	— ml el fl		GENIE-land IGCM-land ENTS Fixed	*Meissner et al.* [2003] *de Forster et al.* [2000] *Williamson et al.* [2006] —
Ice-sheets	gi fi		GLIMMER Fixed	*Payne* (1999)
Vegetation	— — — —		— TRIFFID ENTS Fixed	— *Cox* [1998] *Williamson et al.* [2006] —
Ocean biogeochemistry	bg		BIOGEM	*Ridgwell et al.* [2007a]
Deep-sea sediments	sg		SEDGEM	*Ridgwell and Hargreaves* [2007]
Atmospheric chemistry	ac		ATCHEM	*Ridgwell et al.* [2007a]
Terrestrial weathering	rg		ROKGEM	—

Figure 8: Table summarizing the different GENIE modules.

B cGENIE directory structure

This section describes the directories and files pertinent to configuring and running biogeochemistry experiments in cGENIE.

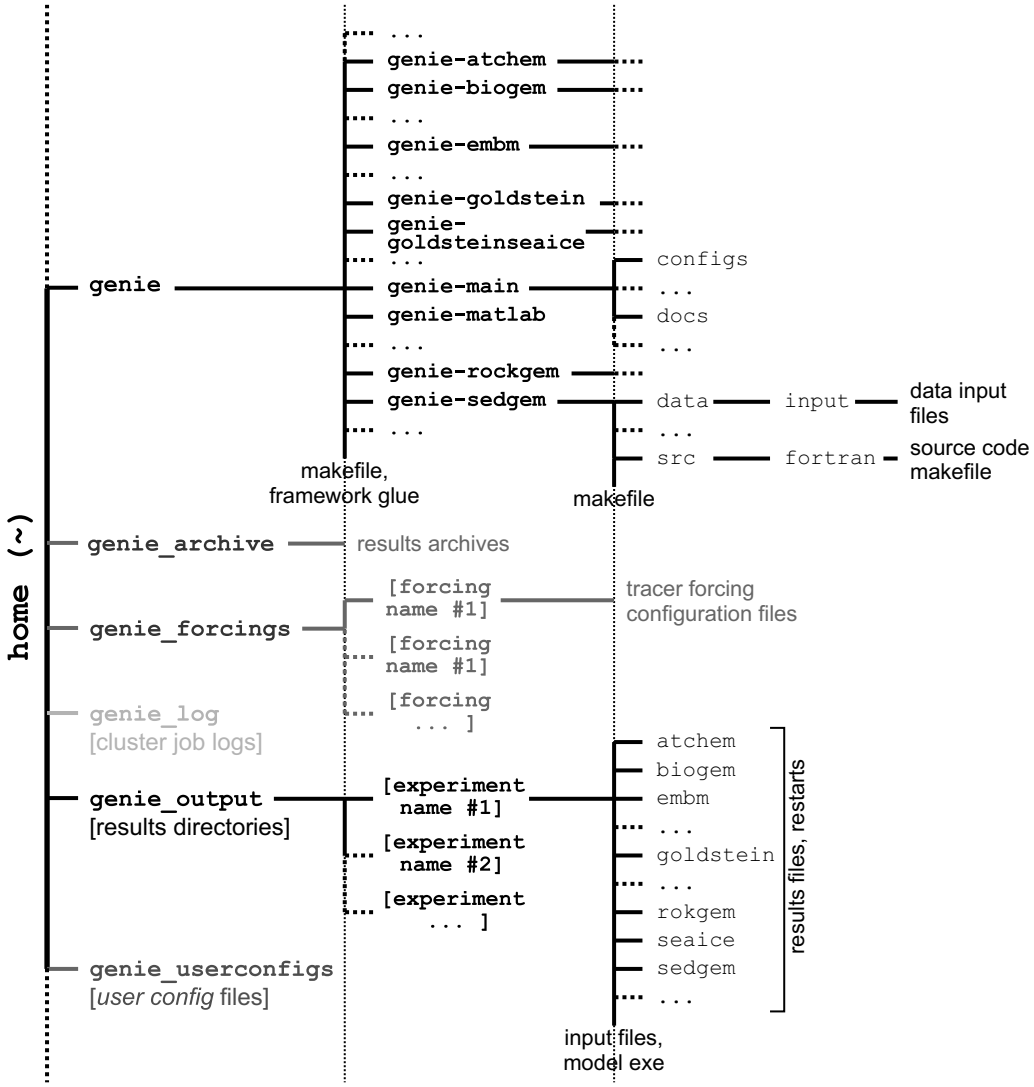

Figure 9: Schematic of the cGENIE file-tree. Note that only the sub-directory structure for the `genie-biogem` module is shown expanded; near-identical sub-structures are present in all the science modules. The dark gray shaded parts of the directory tree are not part of the official GENIE release, but instead required when using the `old_rungenie.sh` script. The light gray branch is required only when submitting cluster jobs exactly as described in this document. The directory in which the genie code tree is installed is assumed to be your home directory and is designated by `~`.

The details of the directory contents are as follows.

- `~/cgenie/genie-atchem/data/input`
  Input data files for the atmospheric chemistry module ATCHEM. Currently there are no data files that are required to be present here.
- `~/cgenie/genie-atchem/src/fortran`
  Source code for the atmospheric chemistry module ATCHEM:
  - `atchem.f90` – Updating of atmospheric composition; re-start file saving.
  - `atchem_box.f90` – [currently just an empty template module - will hold atmospheric 'chemistry' subroutines]
  - `atchem_data.f90` – Load re-start; initialize atmospheric grid and tracers; read in namelist settings.
  - `atchem_lib.f90` – Parameter and array definitions having ATCHEM module scope; set namelist definitions.

- `cpl_comp_atmocn.f90` – Set surface-atmosphere trace-gas interface composition.
- `cpl_flux_ocnatm.f90` – Integrate net trace-gas flux to atmosphere from surface.
- `end_atchem.f90` – [currently does not 'do' anything except print-to-screen]
- `initialise_atchem.f90` – Initialize ATCHEM; read in name-list settings.
- `makefile` – Errr, the makefile ... :o)

- `~/cgenie/genie-biogem/data/input`
  Input data files for the ocean biogeochemistry module BIOGEM. This is the location searched for input data files if the default input directory namelist setting is used, which is set by the namelist parameter: `bg_par_indir_name`. In addition to any files added locally (e.g., for specific model runs) the SVN server holds the following files:
  - Default[48] configuration of tracer forcing of BIOGEM:
    - `configure_forcings_atm.dat` – Selection and configuration of forcing of atmospheric (gas) tracers.
    - `configure_forcings_ocn.dat` – Selection and configuration of forcing of ocean (dissolved) tracers.
    - `configure_forcings_sed.dat` – Selection and configuration of forcing of sediment (particulate matter) tracers.
  - Configuration files for use in the BIOGEM test (`$ make testbiogem`):
    - `genie_eb_go_gs_ac_bg_test_save_sig.dat` – Specification of the times of time-series data saving.
    - `genie_eb_go_gs_ac_bg_test_save_timeslice.dat` – Specification of the times of time-slice data saving.
    - `genie_eb_go_gs_ac_bg_test_windspeed.dat` – Prescribed wind field for calculating air-sea gas exchange.
  - Various illustrative (template) time-series data saving specification files:
    - `save_sig_historical.dat` – Time-series saves specified to cover 1765.5 to 2000.5 at 1.0 year intervals, with 0.5 to 1000.5 following `save_sig_log10.dat` (below).
    - `save_sig_log10.dat` – Time-series saves specified at intervals of 1.0 years up until 10.5 years, then 10.0 years until 100.5 years, etc etc.
    - `save_sig_log10_25kend.dat` – As above, but terminating at 25000.5 (i.e., the mid-point of the year 25000).
    - `save_sig_log10_50kend.dat` – As above, but terminating at 50000.5 (i.e., the mid-point of the year 25000).
  - Various illustrative (template) time-slice data saving specification files:
    - `save_timeslice_historical.dat` – Time-slice saves specified with mid-points at: 1765.5, 1994.5, 2000.5.
    - `save_timeslice_log10.dat` – Time-slice saves specified with mid-points at: 0.5, 10.5, 100.5, 1000.5, etc etc.
    - `save_timeslice_log10_25kend.dat` – As above, but terminating at 25000.5.
    - `save_timeslice_log10_50kend.dat` – As above, but terminating at 50000.5.
  - Modern observed annual mean wind-speed for use as an air-sea gas exchange boundary condition in BIOGEM[49]:
    - `windspeed.dat`

- `~/cgenie/genie-biogem/src/fortran`
  Source code for the ocean biogeochemistry module BIOGEM:
  - `biogem.f90` - Primary ocean biogeochemical cycling subroutine/loop; saving of time-series and time-slice data; re-start file saving.
  - `biogem_box.f90` – Ocean biogeochemical cycling calculations; time-dependent forcing updating; tracer

[48] These are configured for no selected forcings.
[49] The alternative is to estimate wind-speed and hence calculate air-sea gas exchange from the wind-stress field applied to the GOLDSTEIN ocean circulation model. The wind-stress based reconstruction can be chosen by setting: `bg_ctrl_force_windspeed=.false.`

'auditing' calculations.

- `biogem_data.f90` – Load re-start; initialize ocean grid and tracers; ASCII time-series data saving; ASCII time-slice data saving; read in name-list[50].
- `biogem_data_netCDF.f90` – netCDF format time-slice data saving.
- `biogem_lib.f90` – Parameter and array definitions having biogem module scope; set namelist definitions
- `end_biogem.f90` – Final auditing and reporting.
- `initialise_biogem.f90` – Initialize BIOGEM; read in name-list settings.
- `makefile` – Errr, the makefile ... :o)

- `~/cgenie/genie-main` [see: GENIE wiki pages]
- `~/cgenie/genie-main/configs` [see: ...]
- `~/cgenie/genie-main/data/input`
  Input data files required by the cmngem module defining the tracers in the atmosphere, ocean, and sediments and the relationships between the different tracers:
  - `tracer_define.atm`
  - `tracer_define.ocn`
  - `tracer_define.sed`

  The contents of these files must not be altered (nor the files moved, unless there is a very good reason to do so ...).
  Since there never will be a 'very good reason' ... **HANDS OFF**.
- `~/cgenie/genie-main/src/fortran/cmngem`
  Source code for the geochemistry library modules – common routines, constants, and parameters that need to be accessed by all biogeochemistry modules (i.e., ATCHEM, BIOGEM, SEDGEM, and ROKGEM). The source code files are:
  - `gem_carbchem.f90` – Definition and solution of aqueous carbonate chemistry; routines for solving isotopic fractionation; routines for estimating DIC and ALK given other parameters.
  - `gem_cmn.f90` – Parameter and constant definitions.
  - `gem_data.f90` – Read in namelist parameters.
  - `gem_netcdf.f90` – Low-level netCDF file I/O routines.
  - `gem_util.f90` – Miscellaneous routines, such as for reading and writing ASCII file formats, converting between isotopic notations, numerical solution and multi-dimensional linear interpolation, initialization of tracers and inter-tracer relationships
  - initialise_gem.f90 – Tracer property and relationship definition.
- `~/cgenie/genie-matlab`
  MUTLAB scripts for plotting fields from the 2-D and 3-D BIOGEM and SEDGEM netCDF results files, plus related plotting configuration files.
- `~/cgenie/genie-rokgem/data/input`
  ...
- `~/cgenie/genie-rokgem/src/fortran`
  ...
- `~/cgenie/genie-sedgem/data/input`
  Input data files for the (deep-sea) sediment biogeochemistry module SEDGEM. This is the location searched for input data files if the default input directory namelist setting is used, whihc is set by the namelist parameter: `sg_par_indir_name`. In addition to any files added locally (e.g., for specific model runs) the SVN server holds the following files:
  - Masks which specify the sediment grid locations at which sediment cores are generated:
    - `sedgem_save_mask.36x36` – ... .

[50] Would more logically live in `initialise_atchem.f90` ...

    * sedgem_save_mask.72x72 – ... .
  - Bioturbational mixing profile:
    * sedgem_sed_mix_k.dat – ... .
  - Sediment grid topography (bathymetry):
    * sedgem_topo_D.36x36 – ... .
    * sedgem_topo_D.72x72 – ... .
- ~/cgenie/genie-sedgem/src/fortran
  Source code for the ocean sediments biogeochemistry module SEDGEM:
  - cpl_comp_sedocn.f90 – ... .
  - cpl_flux_sedocn.f90 – ... .
  - end_sedgem.f90 – Final auditing and reporting.
  - initialise_sedgem.f90 – Initialize SEDGEM; read in name-list settings.
  - sedgem.f90 – Primary ocean biogeochemical cycling subroutine/loop; saving of time-series and time-slice data; re-start file saving.
  - sedgem_box.f90 – Ocean biogeochemical cycling calculations; time-dependent forcing updating; tracer 'auditing' calculations.
  - sedgem_box_archer1991_sedflx.f90 – ... .
  - sedgem_box_ridgwell2001_sedflx.f90 – ... .
  - sedgem_data.f90 – Load re-start; initialize ocean grid and tracers; ASCII time-series data saving; ASCII time-slice data saving; read in name-list.
  - sedgem_data_netCDF.f90 – netCDF format time-slice data saving.
  - sedgem_lib.f90 – Parameter and array definitions having SEDGEM module scope; set namelist definitions.
  - sedgem_nnutils.f90 – Neural network utilities.
  - makefile – Errr, the makefile ... :o)
- ~/cgenie_archive – This directory IS NOT created automatically for you and is where the old_rungenie.sh run script will attempt to put archived results.
- ~/genie_forcings – This directory IS NOT created automatically for you.
- ~/cgenie_log – The location where standard output and error streams are directed into a file. This directory IS NOT created automatically for you.
- ~/cgenie_output – The default directory where the results will be sent. This directory IS created automatically for you.
- ~/genie_userconfigs

XXX CONTINUE XXX

C Namelist parameter definitions and defaults

Parameters in the model are controlled via *namelists* – lists of parameter names whose values are passed into GENIE when it initializes. The default namelist parameter values are listed in a set of Tables, which can be downloaded from *mygenie.seao2.org*. These are the values and settings that GENIE will use if not told otherwise.

To effect a change in parameter value, the parameter name is simply assigned the new value, either in the *base config* used, i.e. one of the `.config` files in:

`~/genie/genie-main/configs`

Or, more typically, when using the `old_rungenie.sh` model job configuration and submission script: by appending the new namelist assignment(s) to the *user config* file defining the experiment in:

`~/genie_userconfigs`

Either way, the (re-)assignment generally takes the form:

```
namelist = value
```

Where the value is a string, the syntax is:

```
namelist = 'string'
```

The syntax for logical flag (e.g., 'true') assignments is:

```
namelist = .true.
```

For selecting (or de-selecting) tracers, the syntax is:

```
gm_atm_select_nn = .true.
```

for an atmospheric tracer (gas). `nn` is the integer index of the tracer as detailed in the Tables. For ocean and sediment (particulate) tracers, the namelist names take the same form except with `ocn` or `sed` in the namelist parameter name.

If the number of selected tracers in the ocean is changed, so to must the value of `GOLDSTEINNTRACS`, which sets the ocean array dimension and the number of tracers that must be advected, convected, and diffused. The value of `GOLDSTEINNTRACS` must be equal to the number of selected ocean tracers (i.e., the number of `gm_ocn_select_nn` that are `.true.`) *including* temperature (T) and salinity (S). The default value is 2 (just T and S, giving climate-only simulation capabilities), but this value is adjusted in the `.config` files. Any further deviation from the ocean tracer selection requires a new value to be assigned in the *user config*.

For example, for 16 selected ocean tracers (including temperature and salinity), add the line:

```
GOLDSTEINNTRACSOPTS = '$(DEFINE)GOLDSTEINNTRACS=16'
```

to the end of the *user config* file

For setting initial values of tracers, the syntax is:

```
gm_atm_init_nn = 278.0E-6
```

in the case of an atmospheric (gaseous) tracer. Again, `nn` is the index of the tracer. Ocean tracers are initialized similarly[51]. The units of the tracer initialization parameters are given in the summary Tables.

[51] There is no user-configurable initialization of deep-sea sediment composition (in SEDGEM).

D The `runmuffin.sh` script

D.1 Overview

The overall strategy for the `runcgenie.sh` methodology of running *c*GENIE is as follows:

1. Copy a `.config` file[52] defining the required flavor of GENIE plus configuration and calibration of the climatology from `~/genie/genie-main/configs`.
2. Append a series of *namelist* parameter values changes specific to a particular model experiment, particularly those involving the biogeochemistry and carbon cycling such as the specification of a CO2 emissions forcing [53] or selection of feedbacks (e.g. CO2-and-climate, CO2-and-calcification).
   These namelist parameter alternations (compared to the defaults) are provided in the form of a 'patch' file.
   The altered configuration details are added (patched) to create a fine-tuned configuration `*.config` file with a name unique to the model experiment.
   For example[54], taking a basic science module flavor and configuration:
   `genie_eb_go_gs_ac_bg.config`
   and modifying it according to the biogeochem experiment defined by:
   `~/genie_userconfigs/worbe2_preindustrial_1`
   results in the creation of a new configuration file:
   `genie_eb_go_gs_ac_bg.worbe2_preindustrial_1.config`
   The new configuration file will contain additional settings such as:
   - The number of time-steps in the model in order to exactly achieve your requested run length
   - Whether a re-start file is to be used or not, and if so, where the re-start files are located.
   - The location of parameter files required by the various science modules.
3. Finally, the working directory is changed to:
   `~/genie/genie-main`
   and the model is automatically invoked in the 'usual way', i.e.:

   ```
   $ ./genie_example.job -f
         configs/genie_eb_go_gs_ac_bg.worbe2_preindustrial_1.config
   ```

   BUT NOTE that this is to illustrate what the `rungenie.sh` script does automatically for you and `DOES NOT` mean that you should necessarily enter in the `./genie_example.job -f` command by hand (unless you have completely configured an experiment manually).

D.2 technical details

In glorious and wonderful and completely tedious detail, `old_rungenie.sh` actually does the following:

1. ACCEPT PASSED PARAMETERS
   Check that the required parameters (4) have been passed. Set local variables derived from passed parameters.
2. SET LOCAL FILE AND DIRECTORY NAMES
3. CREATE EXPERIMENT CONFIGURATION FILE
   Make a copy of the specified template configuration file (passed parameter #1). Give the configuration file a unique filename by appending the model run ID (passed parameter #3).
4. SET MODEL TIME-STEPPING
   Set up the time-stepping control of model output, data saving, and experiment run length:
   - (a) Set run length in BIOGEM:
     `bg_par_misc_t_start` is the start year, which is assumed to be zero by default.

[52] As outlined earlier, the specification of particular flavors (unique combinations of science modules) are contained in `.config` files with names of the form `genie_aa_oo_ss_xxxx.config`. For instance, a configuration + parameter calibration of the 16-level version of the GOLDSTEIN ocean model together with ocean biogeochemistry, is defined in `genie_eb_go_gs_ac_bg_itfclsd_16l_JH.config`

[53] The details of any biogeochemical forcings are stored in a directory specified by the value of the namelist parameter `bg_par_fordir_name`.

[54] Refer to the tutorial for a detailed description of this particular model experiment.

bg_par_misc_t_runtime is the run length (years) – its value is taken from passed parameter #4.

- (b) Based on the run length, a consistent overall GENIE run length in terms of the number of internal time-steps is set:
  ma_koverall_total, the overall number of internal time-steps in the model, is calculated based on the run length and length of each internal time-step.
  The length of each time-step is determined by the value of ma_genie_timestep (in seconds), and is defined in the .config file by:
  ma_genie_timestep = 365.25*24.0/500 * 3600.0,
  i.e.,
  ma_genie_timestep=63115.2
  giving 500 time-steps per year[55].
  Note that also in .config file: ma_ksic_loop=5 sets the updating of sea-ice to occur every 5 GENIE time-steps, and ma_kocn_loop=5 set the updating of ocean circulation to occur every 5 GENIE time-steps.
  The runtime_defaults.sh value of ma_katm_loop=1 is unchanged, meaning that the atmosphere (the EMBM in this case) is updated each and every GENIE time-step.
  Finally, ma_dt_write sets a default interval of output (in multiples of the GENIE time-step), and is set equal to the total number of time-steps (ma_genie_timestep) to give output writing at the end of the experiment only.
- (c) Set the climate model components' restart saving frequency (iwstp):
  ea_4, go_4, and gs_4 set the frequency of restart saving (multiples of the GENIE time-step). These are set equal to ma_genie_timestep to give re-start saving only at the end of the run.
- (d) Set climate model components 'health check' frequency (npstp):
  ea_3, go_3, and gs_3 set the frequency of 'health check' diagnostics reporting.
  These are set equal to ma_genie_timestep+1, effectively disabling this feature.
- (e) Set climate model components 'time series' frequency (itstp):
  ea_5, go_5, and gs_5 set the frequeny of 'integral diagnostics' reporting.
  These are set equal to ma_genie_timestep+1, effectively disabling this feature.
- (f) Set climate model components 'average' frequency (ianav):
  ea_6, go_6, and gs_6 set the 'output averaging' interval.
  These are set equal to ma_genie_timestep+1, effectively disabling this feature.
- (g) Set ROKGEM terrestrial weathering model component reporting frequency:
  The value of rg_par_screen_output is set equal to ma_genie_timestep+1, effectively disabling this feature.

5. SET CLIMATE MODEL RE-START FILE DETAILS

   - (a) Set netCDF restart saving flag:
     The 'y'/'n' value of ea_31, go_19, and gs_14 determine whether a netCDF format restart file is to be saved. These are set to 'n'.
   - (b) Set ASCII restart output flag:
     The 'y'/'n' value of ea_32, go_20, and gs_15 determine whether an ASCII format restart file is to be saved.
   - (c) Set ASCII 'restart number':
     ea_29, go_17, and gs_12 contain strings used to form the ASCII restart file names.
     The string is appented by .*n*, where *n* is an integer. The value of *n* is incremented at each new re-start save. However, because the frequency of re-start saving has been configured to create only a single restart save at the end of the run, the extension of the restart file will always be .1 when using rungenie.sh (unmodified).

6. CONFIGURE USE OF RESTART
   Configure GENIE to use a restart (the alternative being to start from 'cold').
   Restart namelist items are set according to the presence or absence of the 5th passed parameter (restart experiment name). If the 5th parameter is present when rungenie.sh is invoked, the following action is taken:

[55] For GENIE-1 flavors; a different time-step is used for GENIE-2 flavors.

- (a) Check that the restart experiment (directory) exists. Generate an error message and exit if not.
- (b) Set the climate model components to use a restart, namelist items:
  `ea_7`, `go_7`, `gs_7`
  The syntax is `'c'` for a continuing run (i.e., using a restart), and `'n'` for new (from cold).
- (c) Set the biogeochemistry model components to use a restart, namelist items:
  `ac_ctrl_continuing`, `bg_ctrl_continuing`, `sg_ctrl_continuing`, (`rk_ctrl_continuing`)
  The syntax is LOGICAL (`.true.` or `.false.`, or `t` or `f`)
- (d) Set whether a netCDF restart is used for the climate model components:
  `ea_30`, `go_18`, `gs_13`
  The syntax is `'n'` for no netCDF restart file format (instead it will be ASCII).
- (e) Set the climate model ASCII restart file name namelist values:
  `ea_35`, `go_23`, `gs_18`.
  The name in each case is `'rst.1'` (`rst` being the default saved restart filename, and `.1` indicating the first saved restart (see above).
- (f) Set re-start location (given by optional parameter #5).
  The location of the re-start file for each module is given by a namelist parameter[56]: `xx_rstdir_name` in the case of the climate model modules and where `xx` is the module abbreviation (`ea`, `go`, `gs`), and `yy_par_rstdir_name` in the case of the biogeochemical model modules, with `yy` being the module abbreviation (`ac`, `bg`, `sg`, `rg`).

If restarts are not to be used, then:
`ea_7`, `go_7`, `gs_7` are all set to `'n'`, and
`ac_ctrl_continuing`, `bg_ctrl_continuing`, `sg_ctrl_continuing`, `rk_ctrl_continuing` are all set to `.false.`.

7. APPEND EXPERIMENT SPECIFIC NAMELIST CHANGES
   Append the contents of the user configuration file specified by the run ID (passed parameter #3) and its directory location (passed parameter #2) to the basic flavor and configuration file (parameter #1).
   But first ... in case one has a Windoz user infestation ... the namelist file is conditioned to avoid possibility of carriage return/line-feed screw-ups[57].
8. GO!
   Run the model! Change directory and from `~/genie/genie-main`, invoke:
   ```
   ./genie_example.job -f
           configs/genie_eb_go_gs_ac_bg.worbe2_preindustrial_1.config
   ```
9. CLEAN UP
   Move the configuration file created to the results directory in `genie_output`, and archive entire results directory of the run, by:
   ```
   tar cfz genie_archive/$MODELID.$RUNID.tar.gz $OUTPUTPATH
   ```
   This line will need to be edited if the instaled directory structure differs from the default assumed in this manual. Or this line can simply be commented out (`#`) if not required.

[56] A string describing the full path + filename - see page 44.
[57] This step can be omitted (commented out/deleted) in the event that no Windoz users are involved

E Cluster job submission using SGE

For submitting jobs using Sun Grid Engine (SGE) on a cluster, a basic command[58] 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 *c*GENIE executable must have already been built prior to submitting a job. The easiest way to do this is to run cGENIE 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 job[59]:
  `$ 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 one of the University of Bristol clusters would be:
  ```
  $ qsub -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 queue with the command:

```
$ qstat -f
```

and you can kill a job with the `qdel` command, the job numbers being given by the `qstat` command.

Depending on the cluster setup, it may be possible to graphically check what is going on via the www. In an X window, enter:

```
$ ssh almond.ggy.bris.ac.uk
$ firefox --no-remote &
```

and follow the links: 'Cluster Status (Ganglia)', and then 'Job Queue' to get to the actual queue listing and cluster status.

NOTE: It may be that the FORTRAN compiler is not accessible by the computer nodes. The implication of this is that **the *c*GENIE 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

[58] With 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.

[59] The default priority is 0. A lower priority has a higher value ... !

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.

F FAQ (aka: 'has this dumb question been asked before?')

F.1 Help! My experiment has died ...

If, when using the `old_rungenie.sh` shell script to run a GENIE-1 experiment, it all goes horribly pear-shaped ...

1. The experiment dies absolutely immediately. Check that the `old_rungenie.sh` shell script has executable permissions. Also check that the directory you are trying to run the model from is your home 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 `old_rungenie.sh` shell script.

Other sources of error are due to the configuration of GENIE-1 (or more rarely, due to the model itself):

1. As GENIE 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 '`$HOME`'. Note that as well as making this substitution at the command line, the user config file may also contain instances of '`~`' (such as in specifying particular forcings.
2. A missing/not found error can also arise with some compilers if one of the various ASCII input files to BIOGEM (or SEDGEM) does not have a blank line at the bottom (some vague quirk of the unformatted read used in the FORTRAN code). Check: the user *config file*, and also any boundary condition files being requested, such as a fixed CaCO3:POC rain ratio field (refer to the HOWTO for details about setting a fixed CaCO3:POC rain ratio field).
3. Further trouble can occasionally arise when using Windoz and editing files (e.g., the *user config* file) and it is possible to 'corrupted' the format of the file. For what file(s) you have edited, use the command `dos2unix` to strip off Windoz formatting characters (which are invariably invisible in most editors). The syntax for this (or see the linux Man pages, or even Google it) is
   `$ 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. Running GENIE 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.

F.2 Running and configuring experiments: General

F.2.1 When do I have to recompile GENIE?

You will need to recompile GENIE in the following situations:

1. You have just carried out one of the GENIE tests, e.g., MAKE TEST or MAKE TESTBIOGEM.
2. 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 36x36 to 64x64) or vertically (e.g., going from 8 levels in the ocean to 16).
3. You have changed the number of selected biogeochemical tracers (either gaseous, dissolved, and/or particulate).

The latter two involve a change in compiled array dimension.

The compute nodes of the cluster do not have access to the FORTRAN compiler. Hence, submitted jobs cannot recompile modules and all science modules must be already compiled when a job is submitted.[60]

To recompile, you must first force a clean of the compiled modules. From:

`~/genie/genie-main`

issue:

```
$ make cleanall
```

Now you need to recompile (and re link) the science modules. To do this, 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).

At this point you can 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.)

F.2.2 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 CO2 to a prescribed value (uniformly throughout the atmosphere), the string is '`yyyyz`', as in e.g.: `pyyyyz_RpCO2_Rp13CO2`.

F.2.3 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 GENIE down that may not be absolutely essential to a particular model experiment. These include:

1. The number of tracers - if you don't need 'em, then don't select 'em! Selected tracers are automatically passed to GOLDSTEIN and advected/convected/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 change of a model crash.
2. '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`[61].
3. Time-series results saving. Model tracer (plus some 'physical') properties are bring 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 to option to save a defined sub-set (for example, DIC and PO4 in the ocean but not

[60] It is OK to change the flavor of GENIE as linking is done by the C compiler.
[61] It is `.false.` by default.

ALK). Note that time-series saving of data that is a 2-D average, such as atmospheric composition at the ocean-atmosphere interface, sediment composition at the ocean-sediment interface, or just ocean surface conditions, is less numerically demanding than mean values that have to be derived from a 3-D data field.

4. 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 will all possible data category save namelist flags set to `.true.`). However, note that if you have accidently 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.
5. Alter the degree of asynchronicity between climate and biogeochemistry (see later HOW-TO).

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%[62]
- ADD time-series saving $\Rightarrow$ +15%
- REMOVE ^{13}C isotopic species (= DIC and DOC ocean tracers) $\Rightarrow$ -10%[63]

The basic configuration for a faster 'lego box' cGENIE configuration consists of a 18x18 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:
(a) it's low resolution
(b) taking 48 instead of 96 ocean timesteps per year
(c) BIOGEM is only being updated every 4 rather than every 2 ocean time-steps.
Note: I am still in the process of carrying out numerical stability tests; in the end it may have to slow down a bit.

Because the time-stepping is different a new *rungenie* (make executable) script has to be used:

`runCCSgenie_t48.sh`
(an equivalent ocean-only full carbon cycle including sediments has yet to be created!)
Using the following example *user config*:

```
EXAMPLE_p0000b_SPIN_x1CO2_S18x18
```

and base config:

```
cgenie_eb_go_gs_ac_bg_sg_rg_modern_18x18x8_0i_BASE_t48.config
```

A typical run would look like something like:

```
./runCCSgenie_t48.sh
  cgenie_eb_go_gs_ac_bg_sg_rg_modern_18x18x8_0i_BASE_t48 /
  EXAMPLE_p0000b_SPIN_x1CO2_S18x18 101
```

(Don't forget to do a make cleanall if you were using a different ocean configuration!)
In this configuration 100 years take about 40 seconds, 10 kyr would just take over and hour, and 100 kyr could be run overnight!

[62] Because only a 10 year integration has been carried out with a time-slice saved at 10 years, the computational cost of time-slice saving is disproportionately high as displayed. With a longer integration, the relative cost of saving a time-slice will fall. In constrast, the computational cost as a fraction of total run-time of time-series saving and auditing is likely to remain the same.

[63] The 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).

F.3 Climate

F.3.1 Can I do solar geoengineering (SRM) experiments?

Because of the absence of a dynamical atmosphere, options here are limited. However, some modification of the solar constant, a-la 'giant mirrors in space' is possible. Refer to the relevant HOW-TO on time-dependent modifications of the solar constant.

F.3.2 Can I do carbon dioxide removal (CDR) experiments?

Yes! See HOW-TO.

F.4 Biogeochemistry

F.4.1 In `GEMlite`, does the adaptive step size control work with fixed/prescribed pCO2?

If pCO2 if 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 pCO2 restored during 'normal' time-stepping, pCO2 will change during the accelerated mode if weathering is significantly out of balance with sedimentation. The greater this imbalance, the greater the change in pCO2, and the sooner that time-stepping will be handed back to the normal (full updating) mode.

If you have prescribed changing pCO2, 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 geochemcial gradients in the ocean have been established. (This assumption is broken if CO2 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.

F.4.2 Separate solubility (SST) related changes from stratification and circulation changes

With BIOGEM coupled to the climate model core of GENIE-1 [64], a change in atmospheric CO2 will induce a change in SSTs, which in turn affect the carbon cycle and feedback on CO2 via changes in solubility and via changes in circulation (stratification) and thus biological productivity. There are times when is it helpful to separate out solubility related changes from circulation related changes. This equally applies to dissolved O2 and CO2. 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 '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 CO2 concentration calculated by ATCHEM. 'Turn off' explicit CO2 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 (CO2 = 278 ppm), a value of 2.0 corresponds to the effect of doubling CO2, 4.0 x4 CO2, etc. Altering the value of `ea_20` thus lets you control climate (and circulation) without having to adjust CO2 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 x1 CO2 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 x4CO2, but sets its own SSTs internally rather than use those calculated by the climate model. Setting up the SST

[64] Namelist: `ea_36='y'`

restoring is in principal just like for PO4. The values for the SST field you can simply copy and paste out of a prior x1CO2 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 CO2.

F.4.3 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 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 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. It 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) tolorance[65]:

```
bg_par_misc_audit_relerr = value
```

To halt the model[66] if it fails the tracer drift tolorance:

```
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)
```

[65] By default, this is set to `1.0E-08`.

[66] By default the model will continue running, even if there is an apparent spurious drift in tracer inventories occurring.

```
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)
```

F.4.4 How do I do an ocean CO2 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 CO2 injection – `pyyyyz_FCO2_UNIFORM`. This is provided on mygenie.seao2.org. Download and unpack (`tar xfz pyyyyz_FCO2_UNIFORM.tar.gz`) from the directory: `~/genie_forcings`. As it stands, this is configured to stuff 1 PgC yr-1 of CO2 into the ocean over the course of one year. The location of the CO2 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 form 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 `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 CO2 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 CO2 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 CO2 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 CO2 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).

F.4.5 How can I diagnose changes in the carbon budget due to weathering/sedimentation?

The following example assumes that you are only running with CaCO3 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 respct 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 CaCO3. 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.

F.5 Running experiments: The `almond.ggy.bris.ac.uk` cluster

F.5.1 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 (4) 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 ...

F.5.2 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.

G Trouble shooting

G.1 '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 accidently/somehow been changed. To check for this – do a make cleanall, and then from the cGENIE 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.

G.2 '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 ($CaCO_3$) selected as a particulate tracer in the model, but although when it dissolves it releases Ca^{2+} (and removes Ca^{2+} when $CaCO_3$ is precipitated), you do not have Ca^{2+} 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 Ca^{2+}/Mg^{2+} ratio was very different form today, might you need to select Ca^{2+} (and Mg^{2+}) as an ocean tracer.

H Known issues

H.1 Radiocarbon tolerance

The comparison tolerance for 'make testbiogem' has been raised (01/05/08) from 5 to 19 Ulps (units of last place–think of notches on a ruler marked with numbers resolved by a variable of so many bytes) so that the test will pass when using the g95 compiler (version 0.91). The variables in question is ocn_DIC_14C.

H.2 Ocean tracer number related compilation problems

If the number of dissolved ('ocean') tracers (or the grid resolution) is changed then the model will need to be recompiled from 'clean'. This is because the ocean circulation model (GOLDSTEIN) is compiled with array dimensions sufficient only for the actual number of selected tracers in the ocean (including temperature and salinity). Theoretically, it might be possible to 'clean' only the GOLDSTEIN and BIOGEM modules as well as the interfacing 'glue'. However, it is much safer to request that all modules (libraries, objects, and dependency information) is cleaned up. You can do this from the command line (within `~/genie/genie-main`) by issuing the command:

```
$ make cleanall
```

One symptom of an incorrectly-compiled number of tracers is page-after-page-after-page-after-page- ... of output looking something like:

```
*** WARNING ***
-> Originating location in code [module,subroutine]: gem_carbchem.f90,sub_calc_carb
-> ERROR MESSAGE: Numerical instability at step; 0001 / Data; dum_DIC,dum_ALK,
dum_Ca,dum_PO4tot,dum_SiO2tot,dum_Btot,dum_SO4tot,dum_Ftot,dum_H2Stot,dum_NH4to
t,loc_H4BO4,loc_OH,loc_HPO4,2.0*loc_PO4,loc_H3SiO4,loc_HN3,loc_HS,loc_H,loc_HSO
4,loc_HF,loc_H3PO4,pH(SWS), pHfree, pHtotal, pH (OLD), pH (guess #1), pH (guess
#2),[CO2],[CO32-],[HCO3-]
-> ERROR DATA:      1.801730109436933E-007
-> ERROR DATA:      1.575704792562387E-004
-> ...
```

On the other hand, *appropriate* output from BIOGEM looks like (depending on the exact options set) something rather more like:

```
>>> SAVING BIOGEM TIME-SERIES @ year 0.50  493.088  0.000  14.659  4.479  18.815  34.815  ...
>>> SAVING BIOGEM TIME-SLICE @ year  0.500000000000000
>>> SAVING BIOGEM TIME-SERIES @ year 1.50  707.058  0.000  14.659  4.553  18.831  34.815  ...
```

H.3 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
```

H.4 Re-starts

There is a minor bug associated in how re-starts are done. The shortwave radiation incidence at the ocean surface is used to calculate biological productivity. The spatial field of SW radiation is passed to BIOGEM after the biogeochemical ocean update has been carried out. Although physical boundary conditions such as SW radiation are made available to BIOGEM during initialization, the SW radiation field is only calculated during the time-stepping loop. Thus, the very first time-step that BIOGEM takes has zero in the SW radiation field. This affects experiments from a 'cold' start, but only trivially. More noticeable is that from a re-start, the ocean carbon cycle experiences a brief 'hick-up' with no biological productivity associated with the very first time-step.

This bug was not present in older version of the model, and was introduced associated with a re-ordering of the time-stepping of BIOGEM relative to the climate model components.

I Contact Information

- Andy Ridgwell: andy@seao2.org