GENIE `cupcake` on Windows

Ian Ross

4 March 2015

This document describes how to use the new `cupcake` version of the GENIE model on Windows. Documentation is divided into two sections, one for users of the model and one for those who wish to modify the model.

Throughout this document, shell commands are shown in `typewriter` *font. Commands that extend over several lines are marked with lines ending . . . with the following line beginning . . . .*

1 For GENIE users

1.1 Installation and setup

The following prerequisites are needed in order to install and use GENIE on Windows:

Git You can install Git from here: `http://git-scm.com/download/win`.

Python You need Python version 2.7.9, which can be installed from here: `https://www.python.org/downloads/release/python-279/`.

Microsoft Visual Studio All of the features described here have been tested with *Microsoft Visual Studio Ultimate 2013* (version 12.0.31101.00).

Intel Visual Fortran All of the features described here have been tested with *Intel Parallel Studio XE 2015 Composer Edition for Fortan* (compiler version 15.0.2.179).

I have no way of testing with other versions of Visual Studio or Intel Fortran, but significantly older versions of either are unlikely to work.

To install GENIE, first choose a location for the installation. On Windows, it's probably best just to install GENIE in `C:\cgenie`. In a command prompt window, clone the main `cgenie` repository from GitHub using the command

```
git clone https://github.com/genie-model/cgenie.git
```

This will produce a new directory called `cgenie` containing the model source code and build scripts.

You then need to install NetCDF libraries suitable for use on Windows:

```
c:
cd \cgenie\tools
git clone https://github.com/genie-model/netcdf-min-ifort-win netcdf
```

and you need to add the path to the NetCDF Fortran 90 DLL (`C:\genie\tools\netcdf\ia32\bin`) to your `PATH`.

Before using the model, it's necessary to do a little bit of setup. Go into the new `cgenie` directory and run the `setup-cgenie` script:

```
c:
cd \cgenie
setup-cgenie
```

Note that unlike the Linux setup scripts, this does *not* test for a suitable version of Python – it's up to you to make sure that you have Python 2.7.9 installed! The script will ask where you want to put a number of things – it's usually find to just take the defaults (just hit enter at each of the prompts). In all of what follows below, we'll assume that you chose the defaults. The things the script asks for are:

- The GENIE root installation directory: unless you know what you're doing, accept the `C:\cgenie` default for this.
- The GENIE data directory (default `C:\cgenie-data`) where base and user model configurations are stored, along with forcing files.
- The GENIE test directory (default `C:\cgenie-test`) where GENIE jobs with known good outputs can be stored for use as tests – it's possible to run sets of tests and compare their results with the known good values with a single command, which is useful for making sure that the model is working.
- The GENIE jobs directory (default `C:\cgenie-jobs`) where new GENIE jobs are set up – the `new-job` script (see next section) sets jobs up here by default.
- The default model version to use for running jobs. By default, the most recent released version is selected, but you can type another version if necessary. You can also set jobs up to use any model version later on.

After providing this information, the setup script will ask whether you want to download the data and test repositories. It's usually best to say yes.

Once the data and test repositories have been downloaded, GENIE is ready to use. The setup information is written to a `.cgenierc` file in your home directory (which can be in a number of places on Windows: the `USERPROFILE` environment variable normally points to the right place). If you ever want to set the model up afresh, just remove this file and run the `setup-cgenie` script again.

Important: There are a large number of environment variables that need to be set up for the Intel Fortran compiler to work correctly. You *must* run the GENIE configuration and build scripts from a command prompt window that has these environment variables set. Command prompt shortcuts with the appropriate variables set will be found in the Start menu under: "All Programs" → "Intel Parallel Studio XE 2015" → "Compiler and Performance Libraries" → "Command Prompt with Intel Compiler XE v15.0 Update 2" → "IA-32 Visual Studio 2013 mode". *Note that only 32-bit builds of GENIE are currently supported.*

To check that the installation has been successful and that the model works on your machine, you can run some basic test jobs – in the `C:\cgenie` directory, just type

```
tests run basic
```

This runs an ocean-atmosphere simulation and an ocean biogeochemistry simulation using the default model version selected at setup time.

1.2 Creating new jobs

New GENIE jobs are configured using the `new-job` script in `~/cgenie`. This takes a number of arguments that describe the job to be set up and produces a job directory under `~/cgenie-jobs` containing everything needed to build and run the model with the selected configuration. The `new-job` script should be run as:

```
new-job [options] job-name run-length
```

where `job-name` is the name to be used for the job directory to be created under `C:\cgenie-jobs` and `run-length` is the length of the model run in years. The possible options for `new-job` are as follows (in each case given in both short and long forms where these exist). First, there are three options that control the basic configuration of the model. In most cases, a base and a user configuration should be supplied (options `-b` and `-u`). In some special circumstances, a custom "full" configuration may also be used (the `-c` option).

```
-b BASE_CONFIG   --base-config=BASE_CONFIG
```

The base model configuration to use – these are stored in the `C:\cgenie-data\base-configs` directory.

```
-u USER_CONFIG   --user-config=USER_CONFIG
```

The user model configuration to apply on top of the base configuration – model user configurations are stored in the `C:\cgenie-data\user-configs` directory.

```
  -c CONFIG        --config=CONFIG
```

Full configuration name (this is mostly used for conversions of pre-`cupcake` tests) – full configurations are stored in the `C:\cgenie-data\full-configs` directory.

In addition to the configuration file options, the following additional options may be supplied to `new-job`:

```
  -O     --overwrite
```

Normally, `new-job` will not overwrite any existing job of the requested name. Supplying the `-O` flag causes `new-job` to delete and replace any existing job with the requested name.

```
  -r RESTART    --restart=RESTART
```

One GENIE job can be *restarted* from the end of another. This option allows for a restart job to be specified. This must be a job that has already been run (so that there is output data to use for restarting the model).

```
  --old-restart
```

It may sometimes be useful to restart from an old pre-`cupcake` job. This flag indicates that the job name supplied to the `-r` flag is the name of an old GENIE job whose output can be found in the `C:\cgenie_output` directory.

```
  --t100
```

This flag indicates that the job should use the alternative "T100" timestepping options for the model (i.e. 100 timesteps per year for the default model resolution instead of 96).

```
  -j JOB_DIR     --job-dir=JOB_DIR
```

It can sometimes be useful to put GENIE jobs somewhere other than `C:\cgenie-jobs`. This flag allows an alternative job directory to be specified.

```
  -v MODEL_VERSION    --model-version=MODEL_VERSION
```

Normally, `new-job` will generate a job set up to use the default model version which was selected when the `setup-cgenie` script was run. This flag allows for a different model version to be selected.

Examples

```
  new-job -b cgenie.eb_go_gs_ac_bg.p0650e.NONE ...
            ... -u LABS\LAB_0.snowball snowball 10
```

This configures the first example job in the workshop handout. After running this invocation of `new-job`, a new `~/cgenie-jobs/snowball` job directory will have been created from which the job can be executed.

```
  new-job -b cgenie.eb_go_gs_ac_bg.p0650e.NONE ...
            ... -u LABS\LAB_0.snowball -r snowball snowball2 10
```

This invocation of `new-job` sets up a new `snowball2` job that restarts from the end of the `snowball` job to run for an additional 10 years.

1.3 Running jobs

Once a job has been set up using the `new-job` script, it can be run from the newly created job directory using a "`go`" script. Configuring and running a job is as simple as:

```
c:
cd \cgenie
new-job -b cgenie.eb_go_gs_ac_bg.p0650e.NONE ...
          ... -u LABS/LAB_0.snowball snowball 10
cd \cgenie-jobs\snowball
go run
```

The `go` script has three main options and two advanced options. The basic options are:

`go clean` Remove model output and model executables and compiled object files for the current job setup.

`go build` Compile the required version of the model to run this job – this depends on a number of things, including the selected model resolution, but the build system ensures that model executables are not recompiled unnecessarily.

`go run` Compile the model (if necessary) and run the current job.

Both the `build` and `run` commands can also take a "build type" argument for building debug or profiling versions of the model. For more information about this and about how the build system maintains fresh executables of selected versions of the model, see Section~2. Also see that section for the two "advanced" options to the `go` script, which are used to select alternative "platforms" for a machine – in the normal case, the build system will select the appropriate compilers and flags based on the machine on which the model is being run (assuming that a platform definition has been set up for the machine), but sometimes it may be desirable to select between different compilers on the same machine, for which a `set-platform` option is provided by the `go` script.

1.4 Managing configuration files

Configuration files are all kept in `C:\cgenie-data`, base configurations in the `base-configs` directory and user configurations in `user-configs`. All of this configuration data is held in a Git repository on GitHub, so if you want to add user or base configurations to share with other users, ask someone about how to set yourself up to use GitHub.

1.5 Managing tests

It is possible to save job configurations and results as test jobs with "known good" data. This has two main uses – first, for testing a GENIE installation to make sure that it's working; second, to test that changes to the model don't inadvertently affect simulation results. The second application is of more interest for people changing the GENIE model code, but it can still be useful to save jobs as tests.

The `tests` script in `C:\cgenie` is used to manage and run test jobs. To list the available tests, do

```
tests list
```

and to run an individual test or a set of tests, do

```
tests run <test>
```

where `<test>` is either a single test name (e.g. `basic/biogem`), a set of tests (e.g. `basic`) or `ALL`, which runs *all* available tests. The tests are run as normal GENIE jobs in a subdirectory of `C:\cgenie-jobs` with a name of the form `test-YYYYMMDD-HHMMSS` based on the current date and time. As well as full test job output, build and run logs, a `test.log` file is produced in this test directory, plus a `summary.txt` file giving a simple pass/fail indication for each test.

An existing job can be added as a test using a command like

```
tests add <job-name>
```

where <job-name> is the name of an existing job in C:\cgenie-jobs. Note that you need to run the job before you can add it as a test! The test script will ask you which output files you want to use for comparison for each model component – there are sensible defaults in most cases, but you can select individual files too if you prefer.

There are two other features of the test addition command that can be useful. First, it's possible to give the test a different name than the job it's made from – for example

```
tests add hosing\test-1=hosing-experiment-1
```

adds a test called hosing\test-1 based on the hosing-experiment-1 job. Second, it's possible to say that a new test should be restarted from the output of an existing test. Normally, if a test is created from a job that requires restart files, the restart files are just copied from the job into the new test. Sometimes though, it can be of interest to run the job that generated the restart data, then immediately run a test starting from the output of the first test. This can be done using something like this:

```
tests add foo\test-1=job-1
tests add foo\test-2=job-2 -r foo\test-1
```

This indicates that foo\test-1 is a "normal" test, while foo\test-2 is a test that depends on foo\test-1 for its restart data. When you run a test that depends on another for restart data, the test script deals with making sure that the restart test is run before the test that depends on it. So, for example, you can just say

```
tests run foo\test-2
```

and the test script will figure out that it needs to run foo\test-1 first in order to generate restart data for foo\test-2.

1.6 Managing model versions

For most users, it makes sense to run jobs using the most recent available version of the GENIE model code. This is the option chosen by default when the model is initially set up. However, it can sometimes be useful to run jobs with earlier model versions (or with a development version of the model – see the next section). The GENIE configuration and build system provides a simple mechanism to permit this, hiding most of the (rather complex) details of managing multiple model versions from users.

Model versions are indicated by Git "tags". In order to see a list of available model versions, use the following command in the C:\cgenie directory:

```
git tag -l
```

To configure a job to use a different model version from the default, simply add a -v flag to new-job specifying the model version you want to use. For example, to configure a job to use the cupcake-1.0 version of the model, use something like the following command:

```
new-job -b cgenie.eb_go_gs_ac_bg.p0650e.NONE ...
          ... -u LABS\LAB_0.snowball snowball 10 ...
          ... -v cupcake-1.0
```

Within a job directory, you can see what model version the job was configured with by looking at the contents of the config\model-version file – in non-development cases, this will just contain the Git tag of the model version.

2 For GENIE developers

For developers of GENIE, there are a few extra things to know beyond what's needed to run the model. Most of this is covered in the *GENIE cupcake Configuration and Build System* document.

The only Windows-specific feature to be covered here is how to debug the model in Visual Studio. This is a little complicated and there are a number of possible approaches. The following steps describe the recommended method:

1. Set up a new GENIE job for the conditions you want to use for debugging using the new-job script.

2. Start Microsoft Visual Studio.

3. Open the `C:\cgenie\cgenie-msvs\cgenie-msvs.sln` Visual Studio solution file.
4. Right click on the "`cgenie-msvs`" project line in the Solution Explorer and choose "Properties".
5. In the properties dialog, select "Configuration Properties" → "Fortran" → "Preprocessor" and edit the "Preprocessor definitions" line to include all the coordinate definitions listed in the `config\job.py` file in the job directory of the job you want to use for debugging.
6. Still in the properties dialog, select "Configuration Properties" → "Debugging" and edit the "Working Directory" field to point to the job directory of the job you want to use for debugging.
7. Click "OK" in the properties dialog to save the changed settings.
8. From the "BUILD" menu, select "Rebuild Solution" and wait for the model to be rebuilt.
9. Set breakpoints in source files before starting debugging.
10. From the "DEBUG" menu, select "Start Debugging" and debug as normal. Output from GENIE will appear in a seperate console window.