# Creating 4-D Earth models using GPlates, CitcomS and the Geodynamic Framework

Michael Gurnis, Dan J. Bower, Mark Turner, Ting Yang
Seismological Laboratory, California Institute of Technology, Pasadena, CA 91125, USA
gurnis@gps.caltech.edu, danb@gps.caltech.edu, mturner@gps.caltech.edu, tyang@gps.caltech.edu

Nicolas Flament, Rakib Hassan
Earthbyte Group, School of Geosciences, University of Sydney, NSW 2006, Australia
nicolas.flament@sydney.edu.au, rakib.hassan@sydney.edu.au

August 27, 2019

## Contents

# List of Figures

# 1 Quick Start

This section gives some details to quickly set up and run CitcomS. We will discuss pre-processing and postprocessing.

## 1.1 Installing dependent packages

We can install the required packages and specified versions by running the script *requirements.sh*. This will install the correct versions of Numpy, Scipy, GMT.

Start by running *sudo ./requirements.sh* via the terminal. You will need superuser access.

We will need to pre-process the data before running CitcomS. The pre-processing can be completed tractably on a local machine.

## 1.2 Pre-processing

You will need to create a directory structure detailed in Section 3, and include the relevant data files. Check that there is a valid pid file that ends with the extension "*.cfg".

Edit the pid file, and ensure that the following parameters have the correct:

- datadir
- vel_bound_file (This is in the form: /bvel/bvel.dat)
- lith_age_file (This is in the form: /age/age)
- slab_assim_file (This is in the form: /hist/Hist/)

We also need to edit the *config.cfg* file. The following parameters in *config.cfg* need to point to the correct directories:

- grid_dir = (Insert Data Directory)/grid ; intermediate grids
- hist_dir = (Insert Data Directory)/hist ; history
- ic_dir = (Insert Data Directory)/ic ; initial condition
- ivel_dir = (Insert Data Directory)/ivel ; ivel
- lith_age_dir = (Insert Data Directory)/age ; age
- log_dir = (Insert Data Directory) ; parameters

- ps_dir = (Insert Data Directory)/ps ; postscripts
- trac_dir = (Insert Data Directory)/trac ; tracers
- pid_file = (Insert Data Directory)/*.cfg ; CitcomS pid file
- coord_dir = (Insert Data Directory)/coord ; CitcomS *.coord.* files
- bvel_dir = (Insert Data Directory)/bvel ;

We can then run the *Create_History.py* to generate the pre-processing process. Run the *Create_History.py* with python 3.3, with the address of the *config.cfg* file as the parameter. ie, "python3.3 Create_History.py config.cfg".

During the pre-processing, a out of memory error may occur. This is typically due to the computing of tracers. If freeing up memory and re-running does not work, search for "tracer=1" in the file *pid*.cfg*, and replace it with "tracer=0". This disables the tracers during the pre-processing.

### 1.3 Quick Guide to Running CitcomS on a Cluster

## 2 Introduction

The purpose of this tutorial manual is to describe the workflows and data sets to create 4-D Earth models with GPlates, CitcomS, and the Geodynamic Framework (GDF). This manual assumes familiarity with GPlates, CitcomS, Python and Linux. The underlying concept of the 4-D models is that geophysical data, mostly from plate tectonic reconstructions, are assimilated into a time-dependent model of mantle convection. This data assimilation ensures that the convection model follows the geological constraints captured in the plate reconstructions. As such, the models are a hybrid between dynamic and kinematic approaches. The models are used to make predictions that can be tested with independent data, such as present day seismic structure, regional and global sea level, basin subsidence, heat flow, etc.

There are four aspects of the data assimilation. First, the 4-D simulations are forward models that start at some point in the past and are forward integrated in time to the present day. This means that the initial condition (IC) of the domain must be specified in terms of temperature, and if required, composition. Second, the models conform to a time-dependent velocity boundary condition. This velocity constraint on the top surface is specified as a Dirichlet boundary condition on the flow equation. The velocity boundary conditions can either arise from rigid plate models, or from models with deforming plates. Third, the models can conform to the evolving paleo age of the oceanic lithosphere ("lithosphere assimilation"). Fourth, the structure of subduction zones can conform to a simple thermal model of dipping subducted slabs ("slab assimilation"). The purpose of this last method is to ensure realistic asymmetric dipping subduction that

otherwise can not be obtained with viscous flow, simple rheologies and kinematic boundary conditions. See *Bower et al.* (2015) for a detailed explanation of our progressive data assimilation methods.

The assimilation procedure results in the creation of numerous files and directories over a series of steps. We organize these steps into several main stages:

- Preparing directories and input files with GPlates.
- Creating initial condition (IC) and assimilation (lith, hist, etc.) files.
- Merging assimilation functionality into CitcomS.
- Post processing.

# 3 Preparing Input Directories and Files

In this section we will use a Python script to create the default directory structure to hold the model data, and we will use GPlates to create a set of self-consistent input data.

## 3.1 Creating the Case Directory Structure

The Geodynamic Framework scripts and methods documented in this manual assume a standard directory structure to hold both input and output files for your 4-D model runs.

Typically this directory structure is located on a Unix-like system (Linux or Mac OS) and contained in your working area. Open a terminal window, navigate to where you want to create the case, then run this command:

```
$ create_citcom_case.py sample_case -r 3
```

The string identifier 'sample_case' is the name of your new 4-D Earth model case. The optional arguments '-r 3' will create sub directories for three separate model runs, that will share the same basic kinematic plate reconstruction (i.e. the same GPlates global model).

This script will create the case directory using the case name string, and it will create various subdirectories. Here is a recursive listing of the initial directory structure:

```
$ ls -R sample_case/

sample_case/:
geodynamic_framework_defaults.conf  Reconstruction/  Run-0/  Run-1/  Run-2/
```

```
sample_case/Reconstruction:
Coord/  ICHist/  Topologies/  Tracers/  Velocity/

sample_case/Reconstruction/Coord:

sample_case/Reconstruction/ICHist:

sample_case/Reconstruction/Topologies:

sample_case/Reconstruction/Tracers:

sample_case/Reconstruction/Velocity:

sample_case/Run-0:
geodynamic_framework_defaults.conf@

sample_case/Run-1:
geodynamic_framework_defaults.conf@

sample_case/Run-2:
geodynamic_framework_defaults.conf@
```

Please examine the 'geodynamic_framework_defaults.conf' file. You will see that this file holds path names to system wide, and case specific data. You will edit this file to reflect your particular case and choice of global kinematic model, age grids, etc.

Initially, all of these subdirectories are empty, and as you progress through this manual and build the case, each of the case subirectories will be populated with the necessary input and output files.

Notice that each of the Run-$n$ sub directories contains a soft link to the case top level .conf file.

## 3.2 Geodynamic framework defaults file

Many of the GDF scripts use the "geodynamic_framework_defaults.conf" to reference essential data and paths for the case. The entries in the .conf file are the paths to various data sources, such as the GPlates line data, age grids, etc.

Once you have a copy of the defaults file in your case directory you can update the path names to your chosen data sources. It is critical that the path names are correct because otherwise the scripts will fail,

or worse, produce meaningless output that might be tricky to spot.

## 3.3 Working with Kinematic Models in GPlates

In this section we create three kinds of files with GPlates:

- CitcomS cap feature files, in the GPlates .gpml format.
- Coordinate data of the reconstructed tectonic plate boundaries, in the GMT .xy format
- Surface velocity boundary condition data, in the native CitcomS format.

The coordinate data files are later processed by the geodynamic framework scripts to create several different various forms of the subduction and slab data needed for assimilation. The velocity files are used directly by CitcomS as time-dependent kinematic boundary conditions.

For this step, we work with a specific GPlates reconstruction. The reconstruction needs to contain the specific region and geological evolution you want to model in CitcomS. The GPlates reconstruction also needs to be self-consistent with the data for the paleo age grids within the ocean basins. In this step we do not use the paleo age grids.

We will use Global_Model_WD_Internal_Release_2014.1/ where "WD" stands for "with deformation", indicating this model has deforming regions. This model also contains areas of so-called flat slab subduction. The period of flat slab subduction in North America (the Laramide) starts at 100 Ma in this reconstruction, and we will start this tutorial slightly earlier than 100 Ma to capture the period of onset.

The data files for Global_Model_WD_Internal_Release_2014.1/ are distributed via an SVN server and as a .zip archive. Check out from SVN (or save the data bundle) to your local desktop or other preferred work area. After the check out, or bundle expansion, create the top level directory:

Global_Model_WD_Internal_Release_2014.1/

Using GPlates, we create sub-directories and exported files under this top level working directory. Open GPlates and use the **File** → **Manage Feature Collections** dialog to load all global model files (.rot and .gpml). The globe will be populated and the feature type hierarchy creates a set of new layers. In the Layer Window each type of data has a different color:

- Rotation data : yellow
- Reconstructed Feature Geometries (points, lines, static polygons): green
- Resolved Topological Geometries (closed plate polygons) : pink
- Resolved Topological Networks (deforming zones) : brown

- Calculated Velocity Fields : cyan
- Raster data : red

Check the box next to Global_EarthByte_TPW_GeeK07_2014.1.rot to make it the Default Reconstruction Tree. Next, connect the other rotation files to this primary layer. Open the **Inputs** section, then click on **Add new connection** and select each of the other rotation files (.rot).

Check that the reconstruction has been loaded and is working properly. For clarity, you can hide the regular feature data (green layers). These are the underlying feature data layers that form the plate polygons and deforming zone network topologies. Go to the age of interest (103 Ma for this example) and animate the reconstruction forward in time to present-day (0 Ma). Verify that all of the plate polygons and topological networks close, and make sure there are no spurious lines or artifacts coming and going as a function of time. A clean closure of all of the topological features will demonstrate that all of the rotation data, points, lines polygons, topologies, etc. are all working together properly.

## 3.4 Creating CitcomS Cap Files in GPlates

In this step we create a set of CitcomS mesh cap features representing the surface point locations of the cap. This is necessary to enable the export of files that encode the time-dependent velocity boundary condition at the top surface. The type of mesh (global or regional), and the node density (typically 129 for global models), must match the subsequent 4-D geodynamic models you will run in CitcomS. Global mesh cap files can be generated by GPlates, with the algorithm found in CitcomS.

Regional mesh files must be generated externally with CitcomS, and then converted to .gpml format to be read into GPlates, with "convert_meshes_CitcomS_to_gpml.py" from the geodynamic python framework (an example of this is given below). Future releases of GPlates will alow the creation of regional meshes directly.

This tutorial will use the case of a global mesh, with moderately high resolution of 129 node points per cap side, and we will generate the files from within GPlates.

- Select menu item: **Features** → **Generate Velocity Domain Points** → **CitcomS ...**
- Under Resolution set both **nodeX** and **nodeY** to 129.
- Under Output keep the default **File Name Template**, "%d.mesh.%c"
- Under Output Directory, use the **...** button to open a file dialog.
- On the file dialog navigate to your working copy of Global_Model_WD_Internal_Release_2014.1/ .
- Create a new sub-directory named "Meshes".

- Click OK, and GPlates will create the 12 Mesh cap files in this directory.

Each of these new files will appear in the Layer window as new separate green feature layers. In addition 12 new Velocity layers will be created automatically, and velocity vectors will be plotted on the globe. You can show and hide these points and vectors with the Layer controls, and with the menu items under: **View → Geometry Visibility**

If you already have an existing CitcomS regional mesh (coor.dat), you can easily convert it to .gpml with this command:

```
$ convert_meshes_citcoms_to_gpml.py citcoms_regional coor.dat 0.mesh.0.gpml
```

Be sure to use the name "0.mesh.0.gpml". This is the pattern GPlates recognizes as a valid CitcomS cap file for velocity export.

## 3.5 Exporting Resolved Plate Boundary Data

In this section we will export the reconstructed and resolved plate boundary data as .xy files.

Select the menu item **Reconstruction → Export ...**.

In the Export Menu Dialog check the **Export Time Sequence of Snapshots** button, and adjust the Animate time from 103 Ma to 0 Ma. Keep the default increment of 1 Ma.

Now we will add entries for Resolved Topologies to the Export Data list. Select **Add Export Type** and set the export options in boxes 1 to 4.

- In box 1. Choose Data Type to Export, select **Resolved Topologies (CitcomS specific)** .
- In box 2. Choose Output File, Format Select **GMT (*.xy)**.
- In box 3. Configure Export Options, keep all the defaults set as is.
- In box 4. Specify Output Filenames, keep the default file name Template: **topology_%P_%0.2fMa**.
- Click OK.

You will return to the Main Export Dialog Box.

- Next to Target Directory, use the **...** button to open a file dialog.
- Navigate to your case directory and navigate to "Topologies".

You will return to the Main Export Dialog Box.

- Click **Begin Animation** to export all of the .xy data files.

In the "Topologies/" directory you will find the exported set of files for each age. Here is the list for 0 Ma:

```
[XY]$ ls -1 *_0.00Ma.xy
topology_network_polygons_0.00Ma.xy
topology_network_ridge_transform_boundaries_0.00Ma.xy
topology_network_subduction_boundaries_0.00Ma.xy
topology_network_subduction_boundaries_sL_0.00Ma.xy
topology_network_subduction_boundaries_sR_0.00Ma.xy
topology_platepolygons_0.00Ma.xy
topology_ridge_transform_boundaries_0.00Ma.xy
topology_slab_edges_leading_0.00Ma.xy
topology_slab_edges_leading_sL_0.00Ma.xy
topology_slab_edges_side_0.00Ma.xy
topology_slab_edges_trench_0.00Ma.xy
topology_slab_polygons_0.00Ma.xy
topology_subduction_boundaries_0.00Ma.xy
topology_subduction_boundaries_sL_0.00Ma.xy
topology_subduction_boundaries_sR_0.00Ma.xy
```

These files hold the reconstructed coordinate data for the various feature types. The file name patterns indicate with subset of the total plate model each file contains. The files are composed of header lines starting with the standard GMT delimiter character ">", and coordinate data lines (lon lat).

Some file names have the "sL" or "sR" component. These codes standing for subduction zones with left or right polarity, and are used to plot the position of the subduction zones on a map with "tick marks" on the appropriate side.

The tags "sL" and "sR" are also repeated within the header lines of the feature data blocks of the main files. On the header lines you will also see other data associated with the subduction zone. The header information is used by the geodynamic framework processing scripts to assimilate

The next step is to output the velocity vector data associated with this reconstruction. This data needs to be resolved on a fixed set of points that will be associated with the 4-D models that you will formulate with GPlates. The meshes can either be generated from CitcomS and read into GPlates or, for global models, generated by GPlates itself. For global models, the algorithm used to generate the meshes is identical in both CitcomS and GPlates.

## 3.6 Exporting Surface Velocity Data

Now we will export the surface velocity data in the native CitcomS format.

Select the menu item **Reconstruction** → **Export ...**.

In the Export Menu Dialog Box check the **Export Time Sequence of Snapshots** button, and adjust the Animate time from 103 Ma to 0 Ma. Keep the default increment of 1 Ma.

First, remove any prior export entries in the Export Data list. Next add an entry for Velocities to the list. Select **Add Export Type** and set the export options in boxes 1 to 4.

- In box 1. Choose Data Type to Export, select **Velocities**.
- In box 2. Choose Output File, Format Select **CitcomS global (*)**.

Even if you are processing data for a regional case, still select the CitcomS global option.

- In box 3. Configure Export Options, keep all the defaults set as is. Note that the option exists to smooth boundary velocities.
- In box 4. Specify Output Filenames, keep the default file name Template: **bvel%d.%P**.
- Click OK.

You will return to the Main Export Dialog Box.

- Next to Target Directory, use the **...** button to open a file dialog.
- Navigate to your case sub-directory "Velocity"

You will return to the Main Export Dialog Box.

- Click **Begin Animation** to export the data files.

In the Velocity directory you will find a set files for each age with this file name pattern: "bvel%AGE.%CAP" and "bvel%AGE.%CAP.xy", where AGE is in Ma and CAP varies from 0 to 11. To be invoked by CitcomS directly, You must copy file bvel%MaxAGE.%CAP to bvel%MaxAGE+1.%CAP for each cap number. Here are the lists for 0 Ma:

```
[VELOCITY]$ ls -1 bvel0.?
bvel0.0
bvel0.1
bvel0.2
bvel0.3
bvel0.4
```

```
bvel0.5
bvel0.6
bvel0.7
bvel0.8
bvel0.9
bvel0.10
bvel0.11

[VELOCITY]$ ls -1 bvel0.?.xy
bvel0.0.xy
bvel0.1.xy
bvel0.2.xy
bvel0.3.xy
bvel0.4.xy
bvel0.5.xy
bvel0.6.xy
bvel0.7.xy
bvel0.8.xy
bvel0.9.xy
bvel0.10.xy
bvel0.11.xy
```

The first set of files are the CitcomS boundary conditions. The corresponding .xy files are suitable for plotting in GMT.

# 4 Python pre-processing (input data generation) preliminaries

## 4.1 Software requirements

The python scripts are known to work with the following versions of software. This does not mean they will not work with other versions. However, there is a bug in Generic Mapping Tools (GMT) 4.5.6 that prevents grdfilter from functioning correctly. Therefore, do not use this version of GMT. In addition, you must use a version of GMT4 because GMT5 is not backwards compatible with the scripts.

- Python 3.3.2, 3.3.0

- Numpy 1.7.1
- Scipy 0.12.0
- GMT$\geq$ 4.5.7 and $\leq$ 5

The script "Install_Dependencies.sh" installs the required versions of Software for Ubuntu. This script utilises scripts in the repository https://github.comlikueimo/ubuntu_gmt4, to install dependencies for GMT 4.5.14.

## 4.2 Create_History.py

"Create_History.py" is the master script that is used to generate all of the input data files. This script is actually a wrapper that calls "make_history_for_age.py" multiple times (either in serial or parallel) to produce input data files for the time period of interest (e.g., 100–50 Ma) as specified by the user.

In your working directory, i.e. the location you wish to generate the input data, run:
$ `Create_History.py -e > config.cfg`
to generate a new configuration file (config.cfg) that you can then change. Run the configuration script to generate the corresponding input files using the following command:
$ `Create_History.py config.cfg`

You may want to rename config.cfg but keep the cfg file type ("extension"). For this example, we will call this file config_test.cfg.

## 4.3 Job details and input data

The first section of config.cfg broadly relate to specifying the job, i.e. serial or parallel processing, etc.:

```
#============
# job details
#============
# N.B., keeping with python convention you must capitalize
# True and False for Booleans!

job = smp ; options are 'smp' or 'cluster'.
nproc = 1 ; -1 to use all available procs.
# For 'smp', 'nproc'=1 is the serial case.
# For 'cluster', 'nproc' is ignored.
```

```
jobname = mkhist ; for cluster job only
walltime = 3:00:00 ; for cluster job only
age_start = 1
age_end = 0

DEBUG = False ; generic switch for debugging
VERBOSE = True ; show terminal output

# do not remove processed age and final temperature grids
KEEP_GRIDS = True
PLOT_SUMMARY_POSTSCRIPT = True;

model_name =
```

These defaults setup the script to run in serial mode using only one processor. The two parameters that users will definitely want to change are "age_start" and "age_end". Other parameters:

- DEBUG: delete (False) or retain (True) temporary files after the script has finished running for debugging and development.
- VERBOSE: output real-time information about the script operations to the terminal (True).
- KEEP_GRIDS: delete (False) or retain (True) temporary GMT grid files. This is useful for debugging and development.
- PLOT_SUMMARY_POSTSCRIPT: plot summary postscripts for the temperature and tracer field (True). The postscript files are written the directory specified by "ps_dir":

  ```
  #========================
  # data output directories
  #========================
  # N.B. use an absolute path for mode = parallel

  ps_dir = ./ps/
  ```

  Post script files will be called temp.DEPTHkm.AGEMa.ps (an example is shown in Fig. 1), and they will be concatenated for all depths in pdf files called AGEMa.summary.pdf (this requires pdftoolkit to be installed).
- model_name: prefix for output files (e.g., initial condition, tracer, etc.).

Once the script has finished running, all of the parameters parsed from the user-specified input cfg file

and other parameters used internally by the script are written to [model_name].parameters in the "log_dir" directory:

```
#========================
# data output directories
#========================
# N.B. use an absolute path for mode = parallel

log_dir = ./log/
```

## 4.4 "pid" file, coordinate files, and global and regional models

Regardless of the type of input data you wish to generate, the python script always requires a valid input for pid_file. The script will first attempt to use the information in the pid file to locate the coordinate files that contain the mesh points for each processor. An alternative location for the coordinate files can be given using coord_dir:

```
#===========
# data input
#===========
# N.B. use an absolute path for job = cluster

pid_file = /net/beno2/nobackup1/danb/input/mkhist/test2/pid00000.cfg ; CitcomS pid file

# default coordinate file path is:
#     [datadir]/[proc]/[datafile].coord.[proc]
# or define a user-specified directory to all of the
# [datafile].coord.[proc] files:
coord_dir = /net/beno2/nobackup1/danb/input/mkhist/test2/coord ; CitcomS *.coord.* files
```

You need to run CitcomS for one time step with the correct mesh parameters (i.e., the parameters you will use for your 4-D CitcomS model) to create both the pid file and the coord files. A global CitcomS mesh example is given below, in which radial mesh refinement in an input file is specified by "coor_file":

```
[CitcomS.solver.mesher]
nproc_surf = 12
nprocx = 4
nprocy = 4
nprocz = 2
```

```
nodex = 129
nodey = 129
nodez = 65
coor = 1
coor_file = G5.coor.global.dat
```

The example mesh refinement file (G5.coor.global.dat) is available from the repository `https://svn.gps.caltech.edu/repos/gplates/utils`.

Create_History.py does not distinguish between CitcomS regional (nproc_surf=1) and global (nproc_surf=12) models during most of the processing stages. The various GMT grids are always constructed with a global domain ($R = g$, i.e. 0 to 360 degrees longitude) for simplicity and ease because CitcomS also uses 0 to 360 degrees longitude. During the final data export routines, the data (e.g., temperature) at the nodes defined in the coordinate files (that contain the mesh points for each processor) are written to files.

# 5 Creating Input Data for CitcomS

The first step will be to create a new directory that will have sufficient space to store all of the files to be created, in our example, we will call this directory TEST. Most of the hundreds of files will be stored in a set of subdirectories below this directory.

## 5.1 Age files for "lithosphere assimilation"

In this step, global age files of the lithosphere are mapped from (GMT) age grids to CitcomS-format input data files (ascii). These data files are used to assimilate the temperature of the lithosphere in 4-D Earth models using CitcomS.

First of all, ensure that the following paths and prefixes are set correctly in the "geodynamic_framework_defaults.conf" file that you are using:

```
# path to age grids with (continental) mask
age_grid_mask_dir =
age_grid_mask_prefix = agegrid_final_mask_
```

To create the files for lithosphere assimilation, set

```
#============
# data output
#============
```

```
# thermal age of lithosphere
OUTPUT_LITH_AGE = True
```

in config.cfg. The files [model_name].lith.dat[age] and intermediate grids (if KEEP_GRIDS = True) will be written in the folders:

```
#========================
# data output directories
#========================
# N.B. use an absolute path for mode = parallel

grid_dir = ./grid/
lith_age_dir = ./age/
```

Copies of the age grids are manipulated according to the following parameters in config.cfg (see also Table 1 for a list of all parameters):

```
BUILD_LITHOSPHERE = True ; include an upper thermal boundary layer
UTBL_AGE_GRID = True ; True will use age grids
utbl_age = 300 ; if UTBL_AGE_GRID is False
lith_age_min = 0.01 ; minimum oceanic thermal age
lith_age_max = 300.0 ; maximum oceanic thermal age
# thermal age for non-oceanic regions if CONTINENTAL_TYPES = False
NaN_age = 200.0
```

Ages are mapped from the age grids such that no age is less than lith_age_min and any age with a NaN will be set to NaN_age. Both of these ages are in Ma. The default values for each will be sufficient for most models. Essentially, areas that are continental crust are set to NaN in the paleo age grids, so NaN_age gives the thermal age of the continental lithosphere (except in models where the age of the continents are assimilated, see below).

## 5.2 Thermal initial condition and "slab assimilation" files

In this step, exported GPlates GMT (*.xy) line data created from the topologies and global age files of the lithosphere are mapped from (GMT) age grids to CitcomS-format input data files (ascii). These data files are used to assimilate the shallow portion of slabs in 4-D Earth models using CitcomS. When running these scripts, all of the xy files written by GPlates as well as the paleo age grids consistent with the plate reconstruction (as grd files) must be accessible. Ensure that the following paths and prefixes are set correctly in the "geodynamic_framework_defaults.conf" file that you are using:

```
# path to age grids with (continental) mask
age_grid_mask_dir =
age_grid_mask_prefix = agegrid_final_mask_

# path to age grids without (continental) mask
age_grid_no_mask_dir =
age_grid_no_mask_prefix = agegrid_final_nomask_

# location of gplates exported .xy line data
gplates_line_dir =
```

And for plotting purposes:

```
# location of gplates exported .xy coastline data
# this default is for low resolution coastline data
gplates_coast_dir =
```

To create the initial condition and slab assimilation files, set

```
#============
# data output
#============

# [INITIAL CONDITION]
# temperature initial condition
OUTPUT_TEMP_IC = True

# [HISTORY]
# slab temperature history
OUTPUT_TEMP = True
```

in config.cfg. The files [model_name].hist.dat[age], [model_name].velo.* and intermediate grids (if KEEP_GRIDS = True) will be written in the folders:

```
#========================
# data output directories
#========================
# N.B. use an absolute path for mode = parallel

grid_dir= ./grid/
```

```
hist_dir = ./hist/
ic_dir = ./ic/
```

The important parameters to create slab data for CitcomS in config.cfg are (see also Table 1 for a list of all parameters):

```
BUILD_SLAB = True ; build slabs
radius_of_curvature = 200.0 ; km
# default values for slab dip and depth if GPML_HEADER = False
default_slab_dip = 45.0 ; degrees
default_slab_depth = 500.0 ; km
UM_advection = 1.0 ; non-dim factor
LM_advection = 3.0 ; non-dim factor
vertical_slab_depth = 660.0 ; depth at which to make slabs vertical

# lower thermal boundary layer
BUILD_LTBL  = False ; lower thermal boundary layer
ltbl_age = 300.0 ; age (Ma) of tbl
```

The parameter "default_slab_depth" controls the depth to which slabs are created in the initial condition. The default depth of assimilation is 350 km which defines the 0.5 contour of the slab assimilation stencil (see *Bower et al.*, 2015)). Parameters that modify the assimilation stencil are not visible in the input cfg because most users will not need to change them. The parameter "vertical_slab_depth" controls the depth at which the dip of the slab changes from "default_slab_dip" to vertical. The ratio between "UM_advection" and "LM_advection" determines how much broader the slab is in the lower mantle compared to the upper mantle. The choice of these parameters should be consistent with the mantle viscosity structure used for a given model.

Subduction zones that initiate during the model run or that have an age slightly greater than the age of the initial condition will be created to a depth calculated using the GPML subductionZoneAge property (if defined in the GPlates reconstruction) and the following input cfg parameters:

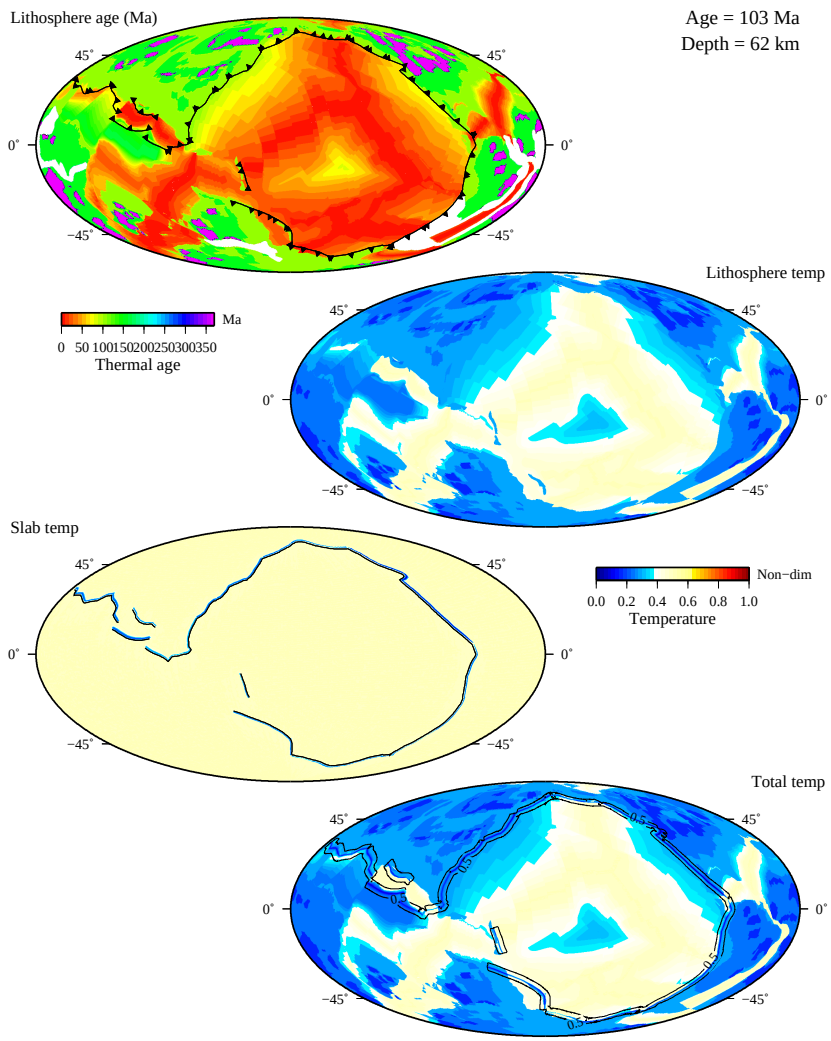
```
# GPML_HEADER must be True for subduction initiation
GPML_HEADER = True ; override defaults with GPML header data
slab_UM_descent_rate = 3.0 ; cm/yr
# from van der Meer et al. (2010)
slab_LM_descent_rate = 1.2 ; cm/yr
```

The parameters "slab_UM_descent_rate" and "slab_LM_descent_rate" have units of cm/yr.

If the reconstructions that you are using contains flat slabs, these can either be included or not using the

flag:

```
FLAT_SLAB = False ; include flat slabs
```




## 5.3 Internal velocity assimilation files

Internal velocity files ensure that the velocity of slabs in the upper mantle is comparable to the velocity of the subducting lithosphere at the trench (see *Bower et al.*, 2015). The following paths and prefixes must be set correctly in the "geodynamic_framework_defaults.conf" file that you are using:

```
# location of gplates exported .xy velocity data (cap, and optionally, gmt)
gplates_velo_xy_dir =
```

Figure 1: *(Previous page.)* Example of `temp.DEPTHkm.AGEMa.ps` from the global case. The top panel show lithospheric ages (no-assimilation areas are in white) and subduction zones. The second panel shows the lithosphere temperature to be mapped to CitcomS. The third panel shows subduction zones as black line and the temperature structure to be mapped to CitcomS. The bottom panel shows the merged lithosphere and slab temperature to be assimilated in CitcomS, as well as the contours of the slab stencil.

```
# location of gplates velocity grids (after processing cap files with grid_maker.py)
gplates_velo_grid_dir =
```

Create_History.py requires that the GPlates velocity grids are named as:
gplates_vx.0.[age].grd
gplates_vy.0.[age].grd
where "vx" is latitudinal velocity and "vy" is longitudinal velocity, both with units of cm/yr. You can create the velocity grids from the output GPlates xy velocity data using the script "grid_maker.py".

In the input cfg file you must also set:

```
#============
# data output
#============

# internal velocity (slab descent)
OUTPUT_IVEL = True

#========================
# data output directories
#========================
# N.B. use an absolute path for mode = parallel

ivel_dir = ./ivel/

# spatial resolution of the prescribed ivel bcs
# levels = 1 ; prescribe at finest mesh
# levels = 2 ; coarsen by 2 in each dimension
# levels = 3 ; coarsen by 4 in each dimension
# for global models you'll likely need levels >= 2
# for regional models try levels = 1 and then increase
# if convergence is poor or solve time is unreasonable
levels = 2
```

"ivel_dir" specifies the location of the output ivel files. "levels" determines if the internal (slab) velocities

are applied to all nodes in the CitcomS computational mesh contained within slabs or if the velocities are applied to a coarser mesh (levels≥1).

## 5.4 Hot blobs and silos to model plumes

Thermal perturbations in the form of spherical blobs or silos (cylindrical base with a hemispherical cap) can be incorporated in the models as seeds for mantle plumes. The spatial location and essential parameters needed to define these perturbations are listed below in the examples. For each category (blob/silo) multiple perturbations can be included by specifying a list of required parameters. The temperature profile within these perturbations can vary with distance, $x$, from the axis of symmetry as:

| Type | Function |
|---|---|
| constant | $A$ |
| exponential | $Aexp(-\frac{x}{r})$ |
| gaussian1 | $Aexp(-\frac{x^2}{r^2})$ |
| gaussian2 | $A(1-\frac{x^2}{r^2})exp(-\frac{x^2}{r^2})$ |

In the functions above, $A$ and $r$ are the user-specified amplitude of a perturbation as nondimensional temperature and the radius of a perturbation, respectively.

Additionally, a mantle adiabat can be incorporated as a linear temperature increase across the mantle as shown below.

```
# thermal blobs
BUILD_BLOB = False ; thermal blobs
blob_center_lon = 50, 130 ; degrees
blob_center_lat = 45, 45 ; degrees colat
blob_center_depth = 2867, 2867 ; km
blob_radius = 200, 400 ; km
blob_birth_age = 230, 220 ; Ma
blob_dT = 0.1, 0.1 ; non-dimensional temperature anomaly
blob_profile = constant, constant ; valid profiles (constant,exponential,gaussian1,gaussian2)

# thermal silos
BUILD_SILO = False ; thermal silos
silo_base_center_lon = 50, 130 ; degrees
silo_base_center_lat = 90, 90 ; degrees colat
silo_base_center_depth = 2867, 2867 ; km
```

```
silo_radius = 200, 400 ; km
silo_cylinder_height= 500, 500 ; km
silo_birth_age = 230, 220 ; Ma
silo_dT = 0.1, 0.1 ; non-dimensional temperature anomaly
silo_profile = constant, constant ; valid profiles (constant,exponential,gaussian1,gaussian2)

# with ADIABAT, temperatures are re-normalized [0,1] at output
# only for extended-Boussinesq or compressible models
BUILD_ADIABAT = False ; linear temp increase across mantle
adiabat_temp_drop = 0.3 ; non-dim w.r.t. super-adiabatic
```

## 5.5 Assimilation of a thermo-chemical continental lithosphere

Compositionally distinct continents can be incorporated in the models, which is particularly relevant to investigate changes in isostatic topography in models based on tectonic reconstructions with deforming plates. We consider the first-order thermal (cold lithosphere) and chemical (light, buoyant lithosphere) characteristics that distinguish continental lithosphere from the convecting mantle. In addition, we consider different continental types, in this example based on the simplified tectonothermal age of the continents (Archean, Proterozoic and Phanerozoic). Continental types may be defined differently depending on the purpose of the models.

### 5.5.1 Exporting xy data from GPlates (continent specific)

#### 5.5.1.1 Exporting continental types

The reconstructed continental types, defined as polygons, must be exported through time. In the current workflow we use the series of files available in:
Global_Model_WD_Internal_Release_2014.1/ StaticGeometries/ContinentalStencils/

```
[ContinentalStencils]$ ls -1
Archean_fixed_poles_extended.gpml
Blocks_crossing_Poles.rot
Phanerozoic_fixed_poles.gpml
Proterozoic_fixed_poles_extended.gpml
```

Load these files in GPlates along with the global rotation file ( Global_EarthByte_TPW_GeeK07_2014.1.rot ); connect the file Blocks_crossing_Poles.rot to the global rotation file (Section 3). These files can then be exported from GPlates as follows:

Select the menu item **Reconstruction** → **Export ...**.

In the Export Menu Dialog check the **Export Time Sequence of Snapshots** button, and adjust the Animate time from 103 Ma to 0 Ma. Keep the default increment of 1 Ma.

Now we will add entries for Reconstructed Geometries to the Export Data list. Select **Add Export Type** and set the export options in boxes 1 to 4.

- Box 1: choose Data Type to Export, select **Reconstructed Geometries**.
- Box 2: choose Output File, Format Select **GMT (*.xy)**.
- Box 3: configure Export Options, untick **Export to a single file**.
- Box 4: specify Output Filenames, keep the default file name Template: **reconstructed_%0.2fMa**.
- Click **OK**.

You will return to the Main Export Dialog Box.

- Next to Target Directory, use the **...** button to open a file dialog.
- Navigate to your working copy of "Continental_types_TEST".

You will return to the Main Export Dialog Box.

- Click **Begin Animation** to export all of the .xy data files.

A series of xy files will be created under three directories in our case.

```
[Continental_types_TEST]$ ls -1
Archean_fixed_poles_extended
Phanerozoic_fixed_poles
Proterozoic_fixed_poles_extended
```

The exported continental types are used to make global age grids with ages for the oceanic and continental lithospheres. Each continental type is assigned a negative integer to distinguish continents from oceans. These integers will later be mapped to a thermal lithospheric thickness. The age grids with oceans and continents are typically called agegrid_final_with_continents_220.grd where 220 is the age in million years.

#### 5.5.1.2 Exporting "no-assimilation" stencils

The age (thickness) of the thermal lithosphere should be assimilated in rigid areas, but not in deforming areas so that the thermal structure can evolve much more naturally through advection–diffusion and the over-riding deforming plate. Therefore, the shape of the deforming plate networks must be exported from

GPlates to be made into no assimilation "stencils". This is straightforward when deformation continues onto the present day where the shape of the deforming areas can be exported directly from the deforming networks:

Select the menu item **Reconstruction** → **Export ...**.

In the Export Menu Dialog check the **Export Time Sequence of Snapshots** button, and adjust the Animate time from 103 Ma to 0 Ma. Keep the default increment of 1 Ma.

Now we will add entries for Resolved Topologies to the Export Data list. Select **Add Export Type** and set the export options in boxes 1 to 4.

- Box 1: choose Data Type to Export, select **Resolved Topologies (CitcomS specific)** .
- Box 2: choose Output File, Format Select **GMT (*.xy)**.
- Box 3: configure Export Options, keep all the defaults set as is.
- Box 4: specify Output Filenames, select only **Export all Network Boundaries to a single file**, keeping the default file name Template: **topology_%P_%0.2fMa**.
- Click OK.

You will return to the Main Export Dialog Box.

- Next to Target Directory, use the **...** button to open a file dialog.
- Create a new directory named for example "No_assimilation_stencils_2014.1".

You will return to the Main Export Dialog Box.

- Click **Begin Animation** to export all of the .xy data files.

When deformation ends in the past, the final geometry of the deforming network must be propagated to younger times. Simply load the file:
Global_Model_WD_Internal_Release_2014.1/ StaticGeometries/Deforming_networks_static_2014.1.gpml along with the global rotation file in GPlates, and export the reconstructed geometries as GMT (*.xy) as when exporting continental types. Merge (concatenate) the obtained .xy files with the .xy files obtained by exporting the outline of the ("dynamic") deforming networks in the proviso step. Save the resulting concatenated files as topology_network_polygons_$AGE.00Ma.xy files in a new directory, for example No_assimilation_stencils_2014.1/Dynamic-Static/.

### 5.5.2 Age files for lithosphere assimilation with continents

The procedure to create age files using make_history_for_age.py for lithosphere assimilation is similar to that described in Section 5.1 with a few additions.

Ensure that the following paths and prefixes are set correctly in the "geodynamic_framework_defaults.conf" file that you are using:

```
# path to age grids with (continental) mask
age_grid_mask_dir =
age_grid_mask_prefix = agegrid_final_mask_

# if using continents, path to age grids with continental ages
age_grid_cont_dir =
age_grid_cont_prefix = agegrid_final_with_continents_

# location of no assimilation stencils (after merging dynamic deforming polygons from GPlates
# and static, "ghost" polygons)
no_ass_dir =
```

In the configuration file for Create_history.py (your copy of config.cfg), set

```
CONTINENTAL_TYPES = True
```

Continental types identified by negative integers are then each assigned an age in order to obtain the desired lithospheric thickness of the thermal lithosphere from the half-space cooling model following $y_L = 2.32\sqrt{\kappa t}$, where $y_L$ is the thickness of the lithosphere, $\kappa$ the thermal diffusivity and $t$ the age of the lithosphere. Note that the ages used in the following do not have a geological meaning: we are simply using the lithospheric assimilation routines to obtain the desired thickness of the thermal lithosphere.

In the example below

```
stencil_values=-4,-3,-2,-1
stencil_ages=104,103,153,369
```

Archean continents (-1) are 250 km thick, Proterozoic (-2) continents are 160 km thick and all other continents are 130 km thick.

To obtain fully dynamic deforming zones, set

```
NO_ASSIM = True
no_ass_age = -1000
no_ass_padding = 100
```

The value no_ass_age should be smaller than -999 (-1000 by default) and the value no_ass_padding (km) creates a no-assimilation buffer zone intended for narrow deforming areas such as passive margins.

### 5.5.3 Buoyant continents using numerical tracers

In this step, a global tracer file is mapped from the (GMT) age grid at the age of the initial condition to CitcomS-format input data files (ascii).

In the configuration file for Create_history.py (your copy of config.cfg), set

```
# tracer initial condition
OUTPUT_TRAC_IC = True
```

The (very large) tracer file model_name.tracer.${AGE}Ma will be written in the folder:

```
#========================
# data output directories
#========================
# N.B. use an absolute path for mode = parallel

trac_dir = ./trac/
```

The number of tracers per element is given by:

```
#=========================
# tracer initial condition
#=========================
# this is approximate, because tracers are uniformly distributed
tracers_per_element = 30
```

The tracers are then mapped to CitcomS from the (GMT) age grids with continents using the stencil_values defined in the previous step (section 5.5.2).

In this example, we have five geological "terrane" types: oceanic, Phanerozoic, Proterozoic, Archean. For simplicity, we have taken the thickness of the chemical lithosphere (defined below by depth_stencil_value_ in km) equal to that of the thermal lithosphere (defined in the previous step by stencil_ages in Ma - stencil ages are converted to depths according to the half-space cooling model).

```
# Build tracer field with continents using 'stencil_values'
# tracer flavors and depths

# for positive thermal ages (i.e., oceanic)
```

```
# note: must be '0' suffix
flavor_stencil_value_0 = 0
depth_stencil_value_0 = 410

# for stencil value -1
flavor_stencil_value_1 = 1,2
depth_stencil_value_1 = 40,250

# for stencil value -2
flavor_stencil_value_2 = 1,3
depth_stencil_value_2 = 40,160

# for stencil value -3
flavor_stencil_value_3 = 1,4
depth_stencil_value_3 = 40,130

# for stencil value -4
flavor_stencil_value_4 = 1,4
depth_stencil_value_4 = 40,130

# etc. for more stencil values, e.g.,
# flavor_stencil_value_5 = 0
# depth_stencil_value_5 = 410
```

In addition, it is desirable to remove continental tracers next to subduction zones for numerical reasons (specifically to limit the entrainment of buoyant tracers to depth). This can be done using the following parameter:

```
# set region around slabs to ambient flavor (0)
SLAB_STENCIL = True
# stencil width: 300 km is consistent with the default width of the thermal stencil
# wide stencils limit crustal thickening along convergent margins
# narrow stencils avoid a gap along convergent margins but may result in significant
# crustal thickening and unrealistic elevations along convergent margins
slab_stencil_width = 300 ; km - suggested range: 100-300 km
```

The postscript files produced by the script will show the continental types used to create tracers (Fig. 4).

A deep layer can also be added to model the evolution of chemical piles by using:

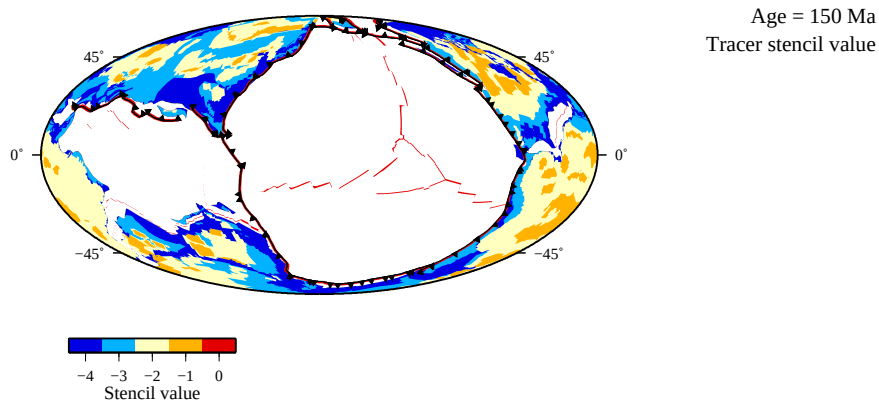



Figure 2: Example of mapped continental types for a global case starting at 150 Ma (`tracer.AGEMa.ps`).

```
# uniform dense layer at base of mantle
DEEP_LAYER_TRACERS = True
deep_layer_thickness = 300 ; km - 113 km gives 2 per cent of Earth's volume.
# flavor should not be 0 (0 is always ambient flavor)
deep_layer_flavor = 5
```

You will want to ensure that deep_layer_flavor is different to the previously defined flavors.

Finally, for numerical efficiency, the option is given to remove tracers for a set depth range:

```
# eliminate tracers between these bounds
# this saves memory when using the hybrid method to compute composition
NO_TRACER_REGION = True
no_tracer_min_depth = 410 ; km
no_tracer_max_depth = 2604 ; km
```

If NO_TRACER_REGION is "True" then "hybrid_method" must also be set to "1" in the input file of the CitcomS run:

```
[CitcomS.solver.tracer]
tracer = on
hybrid_method = 1
```

The "hybrid method" is similar to the "ratio method" (see *Tackley and King*, 2003) to compute composition, except that composition is assumed to be "ambient" (0) if no tracers are present in an element (see Section 6.8 for more details). For certain types of problems, the hybrid method is an advantageous approach because:

- Fewer tracers are required, which saves time in the composition advection routine.
- For most parts of the domain, the preferred statistical characteristics of the ratio method are retained

(effectively, fewer tracers are required in these regions versus the absolute method, see *Tackley and King* (2003)).

Upon running, the script outputs the PS file surface_types.ps that allows the user to check the correct definition of the continental types in map view (Figure 7).

To scale continental elevations, the chemical buoyancy of the different tracers must then be adjusted by iteratively running the initial condition for the final CitcomS run.

## 5.6 Synthetic (regional) models

We develop synthetic regional models in *Bower et al.* (2015) to demonstrate that the slab buoyancy flux is comparable to the flux of the lithosphere at the trench. See Case K, 1–10, in the paper. These models are developed using entirely synthetic data that is constructed using the following parameters in the input cfg:

```
#==============================
# synthetic regional model only
#==============================
SYNTHETIC = False ; master switch for synthetic regional models
OUTPUT_BVEL = True ; output velocity boundary conditions
bvel_dir = ./bvel ; bvel
fi_trench = 0.8 ; radians
TRENCH_CURVING = False
curving_trench_lat = 0.0 ; degrees
subduction_zone_age = 100 ; Ma
plate_velocity = 5 ; cm/yr
plate_velocity_theta = -1 ; direction of velocity (non-dim)
plate_velocity_phi = 1 ; direction of velocity (non-dim)
velocity_smooth = 200 ; bvel smoothing (Gaussian filter)
no_of_edge_nodes_to_zero = 3 ; no of edge nodes to smooth across (x and y)
overriding_age = 50 ; Ma
rollback_start_age = 100 ; Ma
rollback_cm_yr = 0 ; cm/yr (direction is always -phi)
```

## 5.7 Regional models

The difference in the assimilation work flow between regional and global models mainly rest with the generation of surface velocity data. To export surface velocity data files in regional models, you need to follow these several steps:

#### 5.7.0.1 Generate CitcomS coor file

You can use *Make_citcoms_coord_file.py* to generate an uneven (or even if you like) CitcomS mesh file. e.g.

```
Make_citcoms_coord_file.py -v -f Coord.cfg
```

Where Coord.cfg is the controling file you give. Here is an example of the cfg file:

```
name_prefix = Nz65
verbose = True

nproc_surf = 1
nodex = 129
nodey = 129
nodez = 65

degrees_or_radians=degrees
#degrees_or_radians=radians
# Establish the regional domain
#   The map coordinates can either be given as degrees in geographical
#   coordinates (lat, lon) or in radians (colat., long)
lat_min = -35.0
lat_max = 45.0
lon_min = 75.0
lon_max = 155.0
#theta_min=0.8727
#theta_max=2.0944
#fi_min=1.3963
#fi_max=2.6180

radius_outer = 1.0
radius_inner = 0.55
```

```
# Give the refinement details
# if refinement_ratio_theta, refinement_ratio_fi NOT given
# then there will be no refinement in that direction
#
#If degrees_or_radians is set to degrees, above, then these
# are interpreted as degrees
#refinement_ratio_theta = 4.0
#refinement_coor_theta = 30.0
#refinement_width_theta = 5.0
#refinement_ratio_fi = 3.0
#refinement_coor_fi = 275.0
#refinement_width_fi = 10.0

refinement_lower_width=0.005
refinement_lower_ratio=3
refinement_upper_width=0.005
refinement_upper_ratio=6.0
```

*Make_citcoms_coord_file.py* can generate this sample config file with the '-e' command like this:

```
Make_citcoms_coord_file.py -e > Coord.cfg
```

#### 5.7.0.2 Generate gpml mesh file based on the regional CitcomS coor file

To make GPlates able to export regional velocity data, you need to create a gpml format file based on the CitcomS coor file. Run *convert_meshes_citcoms_to_gpml.py* under the directory same as the CitcomS coor files in the following form to generate a gpml file:

```
convert_meshes_citcoms_to_gpml.py format_name in_file out_file
```

Where *format_name* must be set as one of:

**citcoms_regional, citcoms_global or citcoms_regional_cap**

For regional model, *in_file* is name of the citcoms coor file and *out_file* is the prefix of output gpml format file. You must set *out_file* as *0.mesh.0* for regional model to make it readable by GPlates. e.g.

```
convert_meshes_citcoms_to_gpml.py citcoms_regional Nz65.coor.regional.dat 0.mesh.0
```

#### 5.7.0.3 Output velocity file by gplates

Now we will use gplates to export the surface velocity data for CitcomS calculation. First, select the menu item **File** → **Open Feature Collection** to read in the gpml mesh file (Note that GPlates will export regional CitcomS velocity data only if the input .gpml file has exactly the name *0.mesh.0.gpml*).

Second, just as you did in Section 2.5 for the global model, select the menu item **Recinstruction** → **Export** → **Velocities** → **CitcomS global** to export the velocity data. Similar as the global model, the output velocity has the form as:

```
bvel${Age}.0
bvel${Age}.0.xy
```

Finally, different from the global model, this format of data can not be read by CitcomS directly. Regional CitcomS read in velocity data with the format as:

```
bvel${Age}
```

So you need to perform some simple transformation to the velocity file names. Here is one example shell script:

```
#!/bin/sh
cp bvel50.0 bvel51.0
cp bvel50.0.xy bvel51.0.xy
for i in `seq 0 1 52`
do
mv ./bvel${i}.0  ./bvel${i}
mv ./bvel${i}.0.xy  ./bvel${i}.xy
done
```

You need to change the ages in the script based on your own cases.

# 6 Compiling and running CitcomS for data assimilation

## 6.1 Obtain the base code

Parts of the following instructions are taken directly from the CitcomS manual and you should read sections 2.10–2.13.3 of the current (v3.2.0) user manual available from the CIG website `http://www.geodynamics.org/cig/software/CitcomS`.

In addition to the usual system requirements, you will need a handful of additional development tools installed in order to work with the source from the CIG software repository. First, you must have a Subversion client installed. To check, type svn; it should return a usage message.

```
$ svn
Type 'svn help' for usage.
```

For more information on Subversion, visit the Subversion website `http://subversion.apache.org`. Second, you must have the GNU tools Autoconf, Automake, and Libtool installed. To check, enter the following commands:

- $ `autoconf --version`
- $ `automake --version`
- $ `libtoolize --version`

To check out the latest version of CitcomS, use the svn checkout command:

```
$ svn checkout https://geodynamics.org/svn/cig/mc/3D/CitcomS/trunk CitcomS-assim
```

This will create the local directory 'CitcomS-assim' (if it doesn't already exist) and fill it with the latest CitcomS source from the CIG software repository. The CitcomS directory thus created is called a working copy. Use the update command to obtain version 16400:

```
$ cd CitcomS-assim
$ svn up -r16400
```

Unfortunately there is a bug in version 16400 that produces incorrect mechanical boundary conditions. To solve this problem we 'back-out' the following files in the 'lib' directory to version 16022 by issuing the following commands:

```
$ cd lib
$ svn up -r16022 Ggrd_handling.c Full_boundary_conditions.c BC_util.c Topo_gravity.c
Regional_boundary_conditions.c
```

At this stage you now have an operational 'base code'. However, note that ggrd functionality is likely broken in this base code because it was still being developed at this time (by a different user). However, we do not use ggrd functionality for data assimilation.

## 6.2 Patch the base code for data assimilation

**Important:** The data assimilation patches were specifically written for the Pyre version of CitcomS and therefore are not expected to work with the C-version without further development.

There is another svn that contains the modified files that are necessary to patch the base code to provide data assimilation functionality. Use the svn checkout command (note that you will need checkout privileges):

```
$ svn checkout https://svn.gps.caltech.edu/repos/CitcomS CitcomS-patch
```

Manually copy-and-replace the modified files from CitcomS-patch/CitcomS into your working copy of CitcomS-assim. Ensure that you copy the modified files into exactly the same directory structure as they appear in CitcomS-patch/CitcomS.

Below is a complete list of files that should be copied:

- CitcomS/Components/BC.py
- CitcomS/Components/Param.py
- CitcomS/Components/Tracer.py
- CitcomS/Solver/Solver.py
- lib/Advection_diffusion.c
- lib/BC_util.c
- lib/Composition_related.c
- lib/composition_related.h
- lib/Convection.c
- lib/Element_calculations.c
- lib/Full_boundary_conditions.c
- lib/Full_lith_age_read_files.c
- lib/Full_read_input_from_files.c
- lib/Full_solver.c
- lib/global_defs.h
- lib/Instructions.c
- lib/Lith_age.c
- lib/Output.c
- lib/Problem_related.c
- lib/Regional_boundary_conditions.c

- lib/Regional_lith_age_read_files.c
- lib/Regional_read_input_from_files.c
- lib/Regional_solver.c
- lib/solver.h
- lib/Stokes_flow_Incomp.c
- lib/tracer_defs.h
- lib/Tracer_setup.c
- lib/Viscosity_structures.c
- module/bindings.c
- module/misc.c
- module/misc.h
- module/setProperties.c
- m4/cit_python.m4 [1]

In addition, the "deps" directory includes merlin and pythia versions that are currently used for the Citerra installation. If $ `./configure` is unable to download the egg files, try copying and pasting the "deps" directory into your CitcomS-assim directory.

## 6.3 Load modules (Citerra specific)

On Citerra you need to load certain modules in the terminal window each time you want to compile your CitcomS code. You do not need to load the modules to submit jobs to the queue manager; they are only required when compiling the code. I use the intel compiler and mpi implementation:

```
$ module load intel/intel-12 intel/impi
```

You can check the modules are loaded correctly:

```
$ module list
```

[1] m4/cit_python.m4 was changed around August 2011 by the pylith team to fix a bug in Merlin. However, the change disabled the automatic download of Pythia that we require for CitcomS. Therefore it may be necessary to copy-and-replace this file to prevent the code from exiting with the following error when you run $ `./configure`:

```
configure:  Generating Python egg info
checking for egg-related flags...  failed
configure:  error:  cannot scan Python eggs for flags
```

At present, the default Python version on Citerra is 2.4.3, which is suitable for compiling CitcomS with Pyre. According to the CitcomS manual (v3.2.0), "Pyre in CitcomS does not work with Python 2.7 or higher", although I think incompatibilities have also been reported with some versions of Python 2.6.

## 6.4 Compile CitcomS

Navigate to the top level of your CitcomS code (CitcomS-assim) and run the following commands:

- $ `make distclean` [2]
- $ `autoreconf -i --force`
- $ `./configure`
- $ `make`

Note that the assimilation framework requires CitcomS be installed with Pyre (see CitcomS manual).

## 6.5 Change the Scheduler

Most users have a configuration file that contains the scheduler information, although that you can also include the scheduler commands in a regular CitcomS .cfg file.

The following steps assume that `$HOME/.pyre/CitcomS/CitcomS.cfg` exists. If it doesn't, create it. Ensure your `$HOME/.pyre/CitcomS/CitcomS.cfg` looks exactly like this:

```
[CitcomS]
scheduler = pbs

[CitcomS.launcher]
command = /opt/intel/impi/4.0.3.008/bin/mpirun -np ${nodes} -machinefile ${PBS_NODEFILE}

[CitcomS.job]
queue = default
```

There is also a `[CitcomS.pbs]` group, but you shouldn't need to use this unless you wish to specify advanced options with your job submission.

You should now be able to submit and run jobs.

[2] Remove all files made by previous builds.

## 6.6 Monitor jobs (Citerra specific)

Citerra uses the Maui scheduler and Torque resource manager, and the commands are a little different from LSF.

- To view jobs in the queue: $ `showq` or $ `qstat`
- To diagnose a job: $ `checkjob`
- To delete a job: $ `qdel`

## 6.7 Path (bash specific)

If you are only working with one installation of CitcomS-assim, or have a preferred default, then you can add the following lines to your $HOME/.bash_profile so you can call the binary by simply typing $ `CitcomS` rather than the full path name. Change the CitcomS_DIR variable to the path of your CitcomS-assim directory. For example, your modified $HOME/.bash_profile will contain entries similar to these:

```
CitcomS_DIR=cig/CitcomS
export PATH=$HOME/$CitcomS_DIR/bin:$PATH
```

Finally, remember to apply your changes by re-sourcing:
$ `source $HOME/.bash_profile`

## 6.8 Assimilation parameters

CitcomS with assimilation includes several new parameters that can be included in the input model *.cfg file:

- `[CitcomS.solver.ic] mantle_temp`
  set to the exact "temperature_mantle" used in your Create_History.py input cfg file. This is critical for lithosphere assimilation to be implemented correctly!
- `[CitcomS.solver.param] slab_assim`
  set thermal slab assimilation on (1) or off (0)
- `[CitcomS.solver.param] slab_assim_file`
  prefix for thermal slab assimilation files
- `[CitcomS.solver.param] lith_age_depth_function`
  set on (1) or off (0, default). Lithosphere assimilation depth is a function of the lithosphere age

according to the following equation (see Eq. 4–126 of Turcotte and Schubert):

$$\text{Lithosphere assimilation depth} = \max(\text{lith\_age\_depth} \times 2.32 \times (\sqrt{\text{Age} \times \kappa})/R_0, \text{lith\_age\_min}) \quad (1)$$

Note that:

1. `[CitcomS.solver.param] lith_age_depth` is redefined as a prefactor ($\approx 1-2$). By default, with `lith_age_depth_function` turned off, `lith_age_depth` is the actual assimilation depth ($\approx 0.01-0.05$).
2. `[CitcomS.solver.param] lith_age_min` is used to define a minimum lithosphere assimilation depth. Using Eqn. 1, `lith_age_min` = 23.9 (Ma) gives a minimum lithosphere assimilation depth = 0.01.
3. Besides the above, `[CitcomS.solver.param] lith_age_min` is otherwise unused in CitcomS for data assimilation.

- `[CitcomS.solver.param] file_internal_vbcs`
  set internal velocity boundary conditions on (1) or off (0)
- `[CitcomS.solver.param] vel_internal_file`
  prefix for internal velocity boundary condition files
- `[CitcomS.solver.output] output_optional`
  can additionally include `divv, comp_sph, temp_sph, sten_temp, sten_velo, tracer_dens` for the divergence of velocity[3], spherical harmonic expansion of composition[4], spherical harmonic expansion of temperature, thermal stencil, velocity stencil, and tracer density, respectively. The velocity stencil is 1 for nodes that are turned on and 0 for nodes that are turned off.
- `[CitcomS.solver.tracer] hybrid_method`
  this switch activates what we call the "hybrid method" for determining composition. Essentially the method follows the same algorithm as for the "ratio method", only if an element is devoid of tracers the composition of that element is automatically assigned to ambient composition (zero). This method is useful for these reasons: (1) the code does not crash when too few tracers are in an element. (2) input tracer files can be much smaller because the whole domain does not need to be filled with tracers. (3) Advection of the tracers takes less time (because there are less of them).
- `[CitcomS.solver.param] exclude_buoy_above_znode`
  set on (1) or off (0). Off by default.

[3]The convergence criteria for CitcomS is specified as the absolute value of the divergence of the velocity normalized by the magnitude of the velocity. You will need to post-process the divv output to achieve the same quantity because divv is only the divergence of the velocity.

[4]Not extensively tested.

- `[CitcomS.solver.param] exclude_buoy_znode`
  include all buoyancy sources (temperature, composition, phase) at and below this specified znode (must be an integer) in the RHS of the Stokes solver. Buoyancy at znodes greater than the specified znode is explicitly set equal to zero at each time step. The CMB ($r = 0.55$ by default) is at znode=1 and the znodes (always integers) increase towards the surface. Useful for the calculation of dynamic topography.

# 7 Restarting CitcomS for dynamic topography and total topography (restart_citcoms.py)

The script "restart_citcoms.py" is used to build upon a master CitcomS run and create a series of input files for secondary restart runs. These restart runs are used to generate dynamic and total topograhy data. As with most tools in the geodynamic framework this script uses an input configuration file (.cfg), and the script will produce an example .cfg file by using the "-e" option:

```
$ restart_citcoms.py -e
```

This .cfg file is self-documented, and the intent is that you need to change specific parameters to match the master run. You must change the values for master run's original cfg file and original pid file ("master_run_cfg" and "master_run_pid") You must select which ages you want to restart from ("restart_ages"), and select which type of restart you are performing ("restart_type").

The script "restart_citcoms.py" sets up a sub directory for the restart runs, and makes multiple modified copies of the master run's original .cfg file, one copy for each of the ages. It adjusts various parameters in these secondary .cfg files to refect that these are restart runs, not regular runs.

The default parameters and values that are adjusted are contained in the "dynamic_topography_restart_params" variable of the Core_Citcoms.py module. You may include other specific input .cfg parameters you wish to change for the restarts. See the bottom section of the example "restart_citcoms.py" configuration file.

Additonal information is contained in the "restart_citcoms.py" example config file.

## 7.1 Dynamic topography

A Dynamic topography restart also requires these two parameters be set ("lithosphere_depth_DT" and "lithosphere_temperature_DT") in the "restart_citcoms.py" configuration file.

### 7.2 Total topography

A Total topogrphy restart does not requrire any additional settings in the "restart_citcoms.py" configuration file.

# 8 Grid Making (grid_maker.py)

We have developed a script "grid_maker.py" that directly processes CitcomS data and produces GMT4 grids for select time steps (or ages, or run times). It can process both volume data (e.g., temperature, composition, etc.) and surface data (e.g., surface topography, heat flux, etc.). Furthermore, the script can automatically dimensionalize the non-dimensional output from CitcomS to produce datasets that can be compared directly with observations (e.g., surface heat flux). The dimensionalizations are computed using the values in the CitcomS pid file to ensure that the post-processed data is entirely consistent with the model run. For completeness we list the dimensionalizations that the script uses below.

Like most scripts in the geodynamic framework "grid_maker.py" will give a help message if run without any arguments. "grid_maker.py" uses an input configuration file (.cfg), and the script will produce an example .cfg file by using the "-e" option:

```
$ grid_maker.py -e
```

The example file is self-documenting and contains descriptions of the various parameters. There are four main sections to "grid_maker.py"'s input:

- 1. Model coordinate information: "pid_file" and "coord_dir"
- 2. Time values to grid: "time_spec"
- 3. Levels to grid: "level_spec"
- 4. Fields to grid

Each section has example settings for both CitcomS model data and for GPlates velocity data. For any single use of "grid_maker.py" only one data type (CitcomS or GPlates) is used.

In Section 1 you will set the model coordinate information common to both source types using "pid_file" and "coord_dir".

Please NOTE: you may have to edit the "datadir" entry in your pid_file to reflect the proper paths. For example the Citcom model may have been run on a different machine than the machine used for post-processing. Often the datadir entry can be set to a relative path: "datadir=./data/%RANK"

In Section 2 you will choose which output times to grid using "time_spec". A variety of forms for selecting the times are valid (single time, comma separated list, start/stop/step) and you will chose one form. Because of the nature of the computation model the output time in Millions of Years Ago may not match your request exactly. We have algorithms to choose the closest available output timestep and convert that value to Ma. For example you may set "time_spec = 0Ma/30Ma/10Ma" but the closest output times are actually: 7Ma, 20Ma, 29Ma.

In Section 3 you will choose which levels to grid using "level_spec". A variety of forms for selecting the times are valid (single time, comma separated list, start/stop/step) and you will chose one form. For Volume data fields the level_spec may be values from 0 to nodez-1. For Surface (surf) data fields the level_spec must be set to only nodez-1. For Botttom (botm) data fields the level_spec must be set to only 0. Future versions of "grid_maker.py" may automaticall override these settings for Surf and Bottom fields, but for now you must match the correct level(s) to the fields.

In Section 4. you will choose which fields to grid using a set of configuration file sub blocks of this form:

```
[Grid_1]
field = temp
dimensional = True
#blockmedian_I = 0.5
#surface_I = 0.25
#Ll = ...
#Lu = ...
#T = ...
```

Each block sets the gridding parameters for a single field. The first line of the block holds a string indentifier for the field in square backets. The next line has the requrired parameter 'field', and the cooresponding CitcomS field name. The CitcomS field names may be found in the official Citcoms Manual `http://www.geodynamics.org/cig/software/CitcomS/`. See also Core_Citcom.py the global variable field_to_file_map for more detailed field names to be used in Python scripts. As shown above, additional optional values may be set to pass to GMT to fine tune gridding. Additonal information is contained in the "grid_maker.py" example config file.

## 8.1 Heat flux

CitcomS exports non-dimensional heat flux to *.botm.* and *.surf.* files. Dimensionalize the heat flux for either the top or bottom surface using the following expression:

$$\phi = 1000 \cdot k \frac{\Delta T}{R_0} \cdot \phi' \tag{2}$$

where $\phi$ is dimensional heat flux (mW/m$^2$), $k$ is thermal conductivity (W/m·K), $\Delta T$ is temperature drop ($K$), $R_0$ is Earth radius ($m$), and $\phi'$ is non-dimensional heat flux (output from CitcomS).

## 8.2 Dynamic and total topography

### 8.2.1 Scaling of topography

CitcomS exports non-dimensional stress ($\sigma_{rr}$) to *.botm.* and *.surf.* files. To dimensionalize this value we use the same scaling as equation 11 from the CitcomS manual for pressure:

$$P = \frac{\eta_0 \kappa}{R_0^2} P' \tag{3}$$

i.e.,

$$\sigma_{rr} = \frac{\eta_0 \kappa}{R_0^2} \sigma'_{rr} \tag{4}$$

where primes are non-dimensional.

Vertical stress needs to be converted to dynamic topography. We find the equivalent height for a surface to be deflected about it's equilibrium height that gives that same stress.

$$\sigma_{rr} = \Delta \rho g h \tag{5}$$

where $h$ is the equilibrium height, $g$ is gravitational acceleration, and $\Delta\rho$ is the density difference across the top interface. The density difference is computed using the pid file parameters "density" (underlying mantle $\simeq$3300 kg m$^{-3}$) and "density_above" (overlying water, seawater $\simeq$1025 kg m$^{-3}$). Therefore, $\Delta\rho$ is typically $\simeq$ 2275 kg m$^{-3}$.

Combining the two expressions for $\sigma_{rr}$ and re-arranging gives an equation for $h$:

$$h = \frac{1}{\Delta \rho g} \frac{\eta_0 \kappa}{R_0^2} \sigma'_{rr} \tag{6}$$

It is advisable to remove the mean topography since the average of dynamic topography must be zero for the whole Earth.

## 8.3 Recasting Output to a Plate Frame of Reference

The output data from CitcomS is in the fixed mantle frame of reference. Using a combination of tools in GPlates and options in grid_maker.py we can re-cast the output data to the plate frame of reference. This section outlines the steps to perform in GPlates, the grid_maker.py options, and the settings for the Python Script configuration file ("geodynamic_framework_defaults.conf").

The first step is to identify a set of static plate polygons that coorespond to the same global model chosen in Section 2.2 used to create surface velocity conditions for the CitcomS run. This could be a complete set covering the whole globe, or some subset of plates. For example, this file may be used to select continental areas:

Global_Model_WD_Internal_Release_2014.1/

StaticGeometries/StaticPolygons/StaticPolygonsForPlateFrame/StaticPolygonsForPlateFrame2014.1.shp

Now we will create a set of regularly sampled points over the globe. Open creation dialog with menu item:

**Features → Generate Velocity Domain Points → Latitude Longitude ...**

Leave the default extent (the whole globe), and change the number of latitudinal intervals to 180 and longitudinal intervals to 360. Leave the other options and file name template set to the default values.

Select the folder where you want to save the file. It is best to choose one of the Input directories created in Section 2.1, for example "sample_case/Reconstruction/Topologies".

This step will create a single gpml:MeshNode feature with a single multipoint geometry - visually you will see a set of yellow dots covering the globe (yellow because they have a default plate id set to 0).

The next step is to partition this single feature into a set of features, where each new feature is based upon the boundaries of the static polygons. The partitioning will result in a group of features with single multipoint geometries: one-feature-per-static-polygon, with the plate ids assigned from the polygon.

Go to the menu item **Features → Assign Plate IDs**. Select the Static Polygons layer as the partitioning layer, and click Next. Select the lat_lon_velocity_domain_180_360.gpml layer as the layer to be partitioned, and click Next. Other values should be left as defaults, and you may click Apply.

Now you will see several gpml:MeshNode features, each with a single multipoint geometry, and color coded acording to each static polygon's plate id.

If you are using static polygons that do not cover the whole Earth, (e.g. only continents) then any multipoints not located in any static polygon will still have a plate of if 0 (the will remain yellow). These points will all be in one unassigned feature, and it can be easily removed by selecting one of the points and clicking on the trash can in the Current Feature panel on the right side of the globe.

You may futher edit down this new feature collection to include only those particular plates or polygons you want to recast data to. For example, if you are only interested in one plate, delete all the other features except that plate.

Be sure to save this newly partitioned feature collection - it will be highlighted red in the Manage Feature Collection dialog. This file is now ready to be used as an input to the rest of the plate frame change workflow.

Adjust the "geodynamic_framework_defaults.conf" in your case data output directory to point to this file. Set the "frame_change_input_multipoint_filename" parameter to point to the full path of this newly partitioned multipoint file.

Also confirm that the "frame_change_input_rotation_filename" entry points to the full path of the main rotation file of the global model used in Section 2.2.

Now grid_maker.py can automatically recast CitcomS data from the mantle frame to the plate frames with a simple option setting. In the global options area (above all bracket sections) set "make_plate_frame_grid = True".

During the regular production of mantle frame grids, grid_maker.py will also call an auxiliary script "frame_change_pygplates.py" with the correct options to transform the global data set to the plate frames as specified by the mutipoint features found in the "frame_change_input_multipoint_filename" file. These additional grids will have the file name pattern "casename-fieldname-ageMa-depth-plateframe.grd".

## 8.4 Creating input for 4DPlates

4DPlates is a data viewer program developed by Simula Research Laboratory with specific tools to view geodyamic models. To prepare your grids for use in 4DPlates you may run this command:

```
$ Core_Citcom.py -glb *.grd
```

The argument is a list of grid files you wish to display in 4DPlates. The output will be a .glb file with data set acording to conventions and patterns found from the file names:

- the file name will be the same as the case name
- the field label and id will be parsed from the file name component

- the category will be set to "Citcoms/" with the case name appeneded
- the field label will be set to the reconstruction age as set in each file name component
- the ages of appearance and disapearance will be set by the ages in the data files.

The .glb file is then used to pre-process the grid files into tiled files used by 4DPlates. Depending on how your Windows environment is configured, you may eaither double click on the .glb file, or use "Open With ..." and select '4DPlates_builder'. This data will now show up in 4DPlates in the Layers Library, under the "Citcoms" section, with the case name.

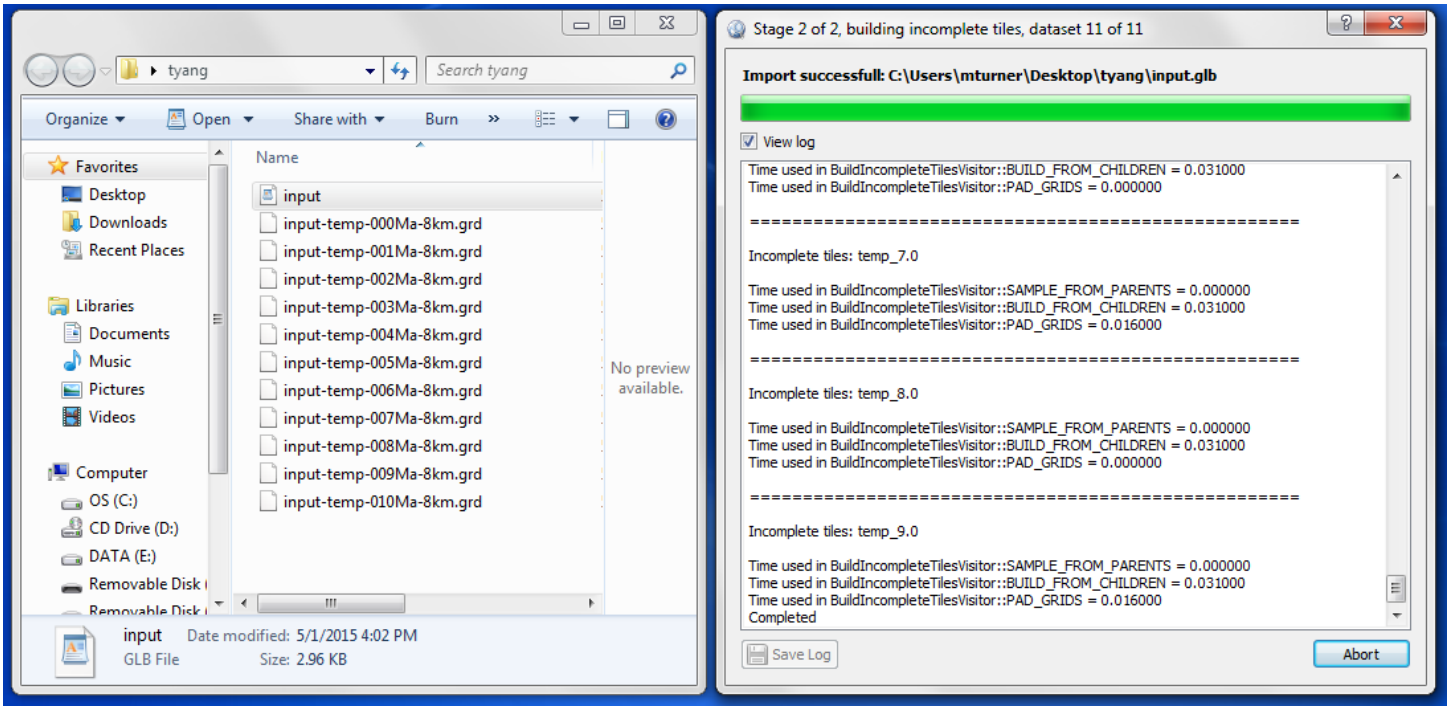



Figure 3: Step 1. Open the .glb file with '4DPlates_builder'.

Please see 4DPlates documentation for more information on its tools and features.

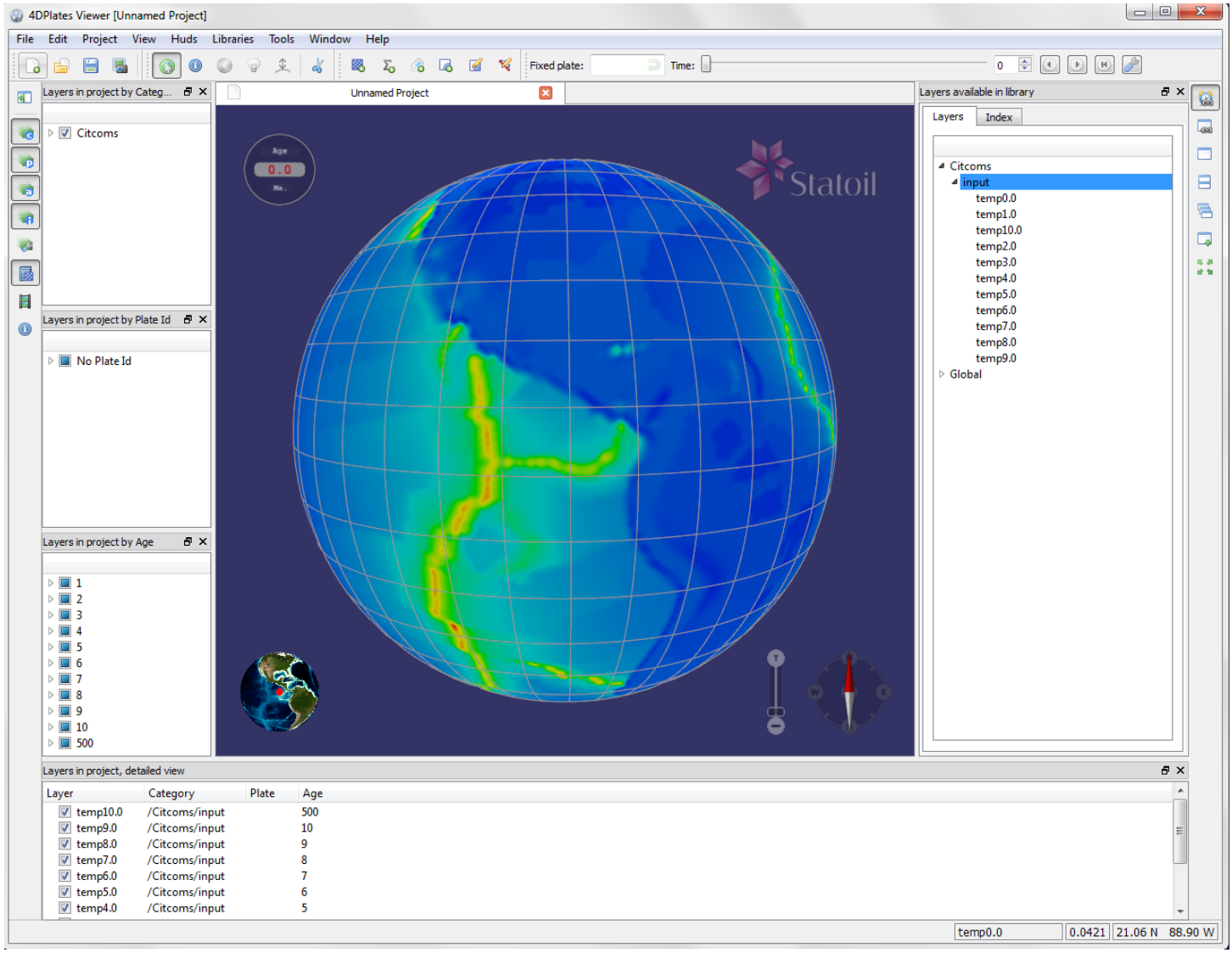



Figure 4: Step 2. Drag the entire CitComS case onto the globe to show time-dependant data.

# References

Bower, D. J., M. Gurnis, and N. Flament (2015), Assimilating lithosphere and slab history in 4-D Earth models, *Phys. Earth Planet. Inter.*, *238*, 8–22, doi:10.1016/j.pepi.2014.10.013.

Tackley, P. J., and S. King (2003), Testing the tracer ratio method for modeling active compositional fields in mantle convection simulations, *Geochem. Geophy. Geosys.*, *4*(4), 8302, doi:10.1029/2001GC000214.

# A Troubleshooting

- Ensure that you have data files for the previous age otherwise CitcomS will exit without providing a helpful error message. For example, if you start a model at 30 Ma, you must also have data files for 31 Ma.

# B Known bugs

- Avoid creating an initial condition using the CitcomS function "lith_age_construct_tic" since it is not compatible with the modifications made for the assimilation workflow. For example, it cannot handle "no assimilation regions" and will truncate lithosphere ages according to `lith_age_min` which could cause problems if also using `lith_age_depth_function`.

# C Appendix 1: Summary Table

- This table sumarizes the standard test case example files described in this document and used to validate the Geodynamic Framework linking GPlates and CitcomS.

| | |
|---|---|
| Section # | Process and purpose |
| Programs: | program, script, and module names |
| Input: | input files and URLs |
| Output: | input files and URLs |

| | |
|---|---|
| Section 0. | ... |
| programs: | ... |
| inputs: | ... |
| outputs: | ... |

| | |
|---|---|
| Section 1. | Display a global plate model |
| Programs: | GPlates |
| Inputs: | URL: https://svn.gps.caltech.edu/repos/GPlates/models/ Global_Model_WD_Internal_Release_2014.1/ |
| Outputs: | On screen; and in ~/work/GPlates/models/ Global_Model_WD_Internal_Release_2014.1/ |
| .../ | raster_0.00Ma.png |

| | |
|---|---|
| Section 2.1 | Export plate boundary line data |
| Programs: | GPlates |
| Inputs: | as Section 1. |
| Outputs: | ~/work/GPlates/models/ Global_Model_WD_Internal_Release_2014.1/ /export/*.* |
| .../ | topology_platepolygons_0.00Ma.xy |
| .../ | topology_ridge_transform_boundaries_0.00Ma.xy |
| .../ | topology_slab_edges_leading_0.00Ma.xy |
| .../ | topology_slab_edges_leading_sL_0.00Ma.xy |
| .../ | topology_slab_edges_side_0.00Ma.xy |
| .../ | topology_slab_edges_trench_0.00Ma.xy |
| .../ | topology_slab_polygons_0.00Ma.xy |
| .../ | topology_subduction_boundaries_0.00Ma.xy |
| .../ | topology_subduction_boundaries_sL_0.00Ma.xy |
| .../ | topology_subduction_boundaries_sR_0.00Ma.xy |

| | |
|---|---|
| Section 2.2 | Export plate and deforming zone velocity data |
| Programs: | GPlates |
| Inputs: | as Section 1. |
| Outputs: | /work/GPlates/models/ Global_Model_WD_Internal_Release_2014.1/ /export/*.* |
| .../ | bvel0.0 |
| .../ | bvel0.1 |
| .../ | bvel0.2 |
| .../ | bvel0.3 |
| .../ | bvel0.4 |
| .../ | bvel0.5 |
| .../ | bvel0.6 |
| .../ | bvel0.7 |
| .../ | bvel0.9 |
| .../ | bvel0.10 |
| .../ | bvel0.11 |

| | |
|---|---|
| Section 4.1. | ... |
| Programs: | Create History.py |
| Inputs: | ... |
| Outputs: | ... |

| | |
|---|---|
| Section 4.2. | ... |
| Programs: | Create_History.py |
| Inputs: | ... |
| Outputs: | ... |

| | |
|---|---|
| Section 4.2. | ... |
| Programs: | Create_History.py |
| Inputs: | ... |
| Outputs: | ... |

| | |
|---|---|
| Section 6. | Restarting CitcomS for dynamic topography and total topography |
| Programs: | restart_citcoms.py |
| Inputs: | restart.cfg |
| Outputs: | A set of secondary CitcomS input .cfg files |

| | |
|---|---|
| Section N. | ... |
| Programs: | ... |
| Inputs: | ... |
| Outputs: | ... |