

Catacomb Reference 1.0

*Components And Tools for Accessible
Computer Modelling in neuroBiology*

Installation

Catacomb is distributed as a single file, `cemb.jar`. It requires java version 1.1.7 or later to run. Once java or `jre` is installed and in the path it can run under unix or from a windows dos prompt with:

```
jre -classpath "cemb.jar" catacomb.modeler
```

Under unix you can also put the `cemb.jar` file in the CLASSPATH variable with:

```
CLASSPATH="cemb.jar"; export CLASSPATH          or  
setenv CLASSPATH "cemb.jar"                    and run  
jre catacomb.modeler                             (or java)
```

The upgrade option from the main file menu checks for a more recent version and downloads it if required. The file `cemb.jar` contains various other executables, including an `appletviewer` for viewing the examples as compound applets. To run it, replace 'modeler' with 'appletviewer' above.

Command line options

By default, java limits its own memory usage. To extend the allocation add the `-mx` flag on the command line. For example `jre -mx128M` allows catacomb to use up to 128Mb of RAM.

If you start catacomb with a further argument it first attempts to load a file of that name. If no file is found and the name matches the name of a model framework it makes a new instance. Failing that the argument is ignored.

Model structure

Catacomb models are trees of objects and lists. Each element of an object may be:

- another catacomb object
- a lists of catacomb objects all of the same type
- a reference to one or more elements of a list defined elsewhere in the tree
- a one- or two- dimensional array of integers or doubles
- an individual variable
- a reference to another object in the tree
- a reference to a field of another object

Variables are booleans, integers, doubles or strings. References to lists may select just one element, a subset, or variable quantities of each element of the whole list or of a subset. Arbitrary references to other objects or their fields are used very rarely.

The catacomb environment contains a permanent *system* component which contains tools for sensitivity analysis and parameter optimisation, recording sessions, building applets and interacting with the JPython interpreter.

Graphs and diagrams

Graphs and diagrams can be rescaled with the mouse alone. Where control points are present which light up when the mouse is over them, these take precedence over rescaling functions.

Other principal mouse operations are:

left click zoom in keeping the point under the mouse fixed

right click zoom out keeping the point under the mouse fixed

long left click zoom out as above

left drag translate so the point on the data where the mouse was pressed moves to where it is released

right drag translation with continuous echo

left drag away and back start box selection. When the mouse re-crosses its starting point it initiates a box cursor. On release this region is mapped to the whole data area.

left drag off the window initiate continuous zooming. Draw clockwise loops to zoom in; anti-clockwise ones to zoom out.

Mouse actions to the left or below the axes scale only the y- or x- axes respectively.

Scrollbars which indicate the fraction of the data visible in the current window. They may dragged at the center to pan across the data or stretched by dragging the ends to change the visible range.

The small menu at the upper right provides shortcuts to several common operations. The capitalized letters indicate single key presses which have the same effect. For example, put the mouse on a graph and press "f" to center it.

Sliders

Drag the knob to set the value. Dragging the mouse up or down away from the slider and then left or right gives finer control.

To change the upper or lower limit click and drag the mouse on the background over the number.

Small squares above and below the arrows change the ranges by fixed amounts, keeping the relative position of the current value fixed:

upper right halve the total range

lower right double the total range

lower left reset to default range

To enter a number by hand, press `return` with the mouse over the slider, type the number and press `return` again.

To change the display style, press the **spacebar** to get checkboxes labelled:

z add a zero button to set the value exactly to zero

p allow setting the number of significant figures

log use a logarithmic instead of linear scale

When the precision button is displayed use left and right mouse clicks to reduce or increase the number of significant figures.

Model browser

The default view of the top level catacomb object shows the tree structure of the permanent components and the current model. Optionally each object, list, and field is shown by a colored label:

orange catacomb object

green list of catacomb objects

blue catacomb object with calculation

mauve reference to a list defined elsewhere

purple reference to a catacomb object defined elsewhere

grey elementary field: boolean, int, double or string

Blue lines connect parents with their children. Red lines show the target list of list references.

The display can be controlled with the mouse as for all graphs. Each label also responds to mouse actions:

left click open a new window to edit the object

right press bring up a menu to alter the display of children or fields, save a single component to a file, or copy it for drag-and-drop operations.

Running models

Objects which define calculations, as opposed to specifying the structure or parameters of a model are shown in blue in the *model browser*. Opening a window on one of these objects initiates the computation and opens another window for their results. For small models with quick calculations, the results are recomputed and displayed whenever a parameter is changed. Larger and slower models have **rerun** buttons to prompt recalculation.

Models may also be run from the JPython interpreter as described in the *scripting* section.

Results display

Each blue calculation object has its own standard results object and may specify further objects for displaying the results differently, each with their own user interface component.

Output options are provided by the default results component under its **data** menu. In particular, the **edit config** option gives access to parameters controlling the display style as an alternative to mouse driven rescaling of the graph.

Online help

Each window has a small "?" button at the top right which loads the online documentation for the object displayed in the window or for the GUI component itself.

All catacomb documentation is accessible this way except this reference card and the FAQ (*May 2000 - almost all, but not quite*).

Saving and restoring models

Models are saved as plain text files, containing declarations of the values of each field by name. Catacomb uses extension `.cm` to identify files containing its own models. Options under the file menu in the *model browser* include

save saves all quantities used to define a model

save config additionally saves positions and ranges of open windows

import reads a single component from a file and tries to find a place for it in the current model.

The *save* option from the tear-off menu of an item in the *model browser* tree saves only that component and its children.

The *DataClipboard* and *ResultsEditor* can save and restore a variety of text and binary data files.

Making applets

The permanent *system* component includes a list of *AppletConfig* instances. A new instance should be made for each applet. The *AppletConfigEditor* extracts panels from currently visible windows and allows them to be laid out in a single panel with accompanying text description.

When an applet object is reloaded, the text appears in the help tool, and the panels are grouped together as specified in one large frame.

When a model with applet definitions is saved, two template html files are written as well as the model (`.cm`) file. One contains an applet tag for loading the applet into a web page, the other a javascript link for opening a new browser window of the right size containing only the applet. To run the applet from a web browser all three files must be in the same directory along with the catacomb.jar archive `comb.jar`.

Drag and drop

Objects and individual parameters may be picked up with the copy option of the right button menu on the corresponding item in the *model browser* or from the *clipboard*. Objects are dropped by clicking the right button.

Dropping data from the clipboard on a graph causes it to be drawn in the background before the normal contents.

Dropping clipboard data on the *waveformEditor* causes it to be converted to a command waveform.

Objects may also be dropped on variable pointers or object pointers to set their targets. Only objects of the appropriate type will be accepted.

Problem domains

Catacomb 1.0 contains frameworks for computing: reaction kinetics; plane and rotationally symmetric reaction diffusion systems; Hodgkin Huxley ion channel models; kinetic scheme ion channel models; passive conduction in branched cells; integrate and fire networks; neuron growth models.

The config object

Parameters in the *config* object control how the user interface behaves.

autooptimize If true, any empty list that occurs is given a new object of the appropriate type.

debugLevel At a debug level of zero, only errors are reported. Higher levels up to 10 provide more information. The case *debugLevel* = 3 causes all internal events to be logged.

fieldVisLevel Fields are only shown by the user interface if their visibility level is lower than the global *fieldVisLevel*. Field visibility is set from the util menu on any component.

The object fields, *help*, *entcolor*, *fieldconf* et al. are used as temporary containers for providing information to or getting responses from the user.

Clipboard

Data loaded from files arrives on the clipboard from where it can be exported to other components. Each set of lines is wrapped in an *LineSet* object, which has an interface allowing colors to be changed, lines to be removed, and the data to be *sparsified* by removing most of the points. The latter is particularly useful for improving performance when densely sampled data is imported of which only the general trends need to be plotted.

Session recording

By default catacomb contains a single *RecordedSession* object independent of any loaded model. This can record and play back all actions accessible through the user interface, with the exception of session recording. On playback models are run in response to parameter changes as usual.

This component is often broken by changes in the software. A longer but more reliable route to making "hands off" demonstrations is via the JPython scripting facility.

Parameter optimisation

The permanent *paramOptimisation* object includes objects for comparing the results of calculations and showing the sensitivity of the results to parameter changes.

Scripting

Catacomb objects and methods can be made accessible to the JPython (*www.jpython.org*) high level interpreted object oriented language simply by ensuring the file `comb.jar` is in the CLASSPATH environment variable when JPython is started. Within JPython,

```
from catacomb import *
accessible by short names (without catacomb.prefix)
cmb = catacomb("cmb")
cmb.loadData("numerics") loads the numerics framework
calc = cmb.getObjectByName("diffEq") finds the equation
calculator object within the model
cmb.editObject("diffEq") starts an editor window for it
cmb.editObject("cmb") starts the main model browser
calc.xmax prints the value of the xmax variable in calc
calc.xmax=20 sets the value of xmax to 20
calc.sync() tells catacomb to synchronise the user
interface with the model data
calc.flagDataChange() emits a dataChange event from the
calc object just as if the variable had been set through a slider
After calling sync() user interface components generally only
indicate that they have caught up with the new value when
approached by the mouse.
To prompt an update in a results display after changing display
parameters from jpython call flagViseChange() on it.
To introduce delays in demonstration scripts:
from java.lang import Thread
Thread.sleep(1000) causes the script to pause for 1000ms
For running in batch mode, it can be convenient to do all recal-
culation from the script instead of in response to dataChange
events:
cmb.setBatchMode() cancels all event propagation
calc.batchRun() runs the calculation
cmb.setInteractiveMode() restores event propagation
```

Reporting problems

The *feedback* option under the *model browser* file menu leads to a feedback form allowing direct submission of comments, questions and bug reports to the catacomb home site. Models demonstrating a problem can be attached to the form. The *save profile* button records user information in a file in the startup directory which is loaded automatically when the feedback form is used.

Lists of frequently asked questions and of known bugs are available on the home site: *nuon.physiol.ucl.ac.uk*.

Copyright © 2000 Robert C. Cannon
Reference card for Catacomb version 1.0

Permission is granted to make and distribute copies of this card provided the copyright notice and this permission notice are preserved on all copies.