Plotting and Analysis for Neural Database-Oriented Research Applications (PANDORA) Toolbox — User's and Programmer's Manual

Cengiz Günay

Document Revision 1000, October 13, 2008

Contents

1 Introduction

1.1 What is the PANDORA Toolbox?

The PANDORA Toolbox is a software package which consists of a collection of MATLAB object-oriented classes and script functions for creating, analyzing and visualizing databases based on data from electrophysiological neuron simulations and recordings.

1.2 Why did you make it?

Motivations to create this software were:

- Analyze data generated by brute-force and other parameter search methods.
- Analyze subsets of parameter spaces and special cases.
- Evaluate robustness of model neurons.
- Find functional roles of specific conductances.

1.3 How is it implemented?

A custom database management system (DBMS) is written from scratch in the MATLAB language. The toolbox design follows object-oriented programming principles. It uses functions from the statistics and signal processing toolboxes of MATLAB, but they are not strictly necessary. It does not use MATLAB's database (DB) toolbox.[1]

1.4 How can I use it?

The PANDORA Toolbox uses an object-oriented approach to provide maximal flexibility for interactive use on the MATLAB command-line.[2] Objects can be created, modified, analyzed, and visualized interactively in few steps. It is straightforward to save and load binary representations of these objects into files. Scripts can be made to programmatically repeat these procedures. Existing object classes are designed with the prospect of future extension, to accommodate new types of data and analyses.

[1] At the time of initial design, the author did not have access to the DB toolbox. Future versions may support the DB toolbox.

[2] This version of the toolbox does not yet have a general graphical user interface (GUI). The author prefers to have a flexible command-line interface than to maintain a limited GUI. However, once commonly used functions can be conveniently placed within a GUI, it will be added to the toolbox.

1.5 Who is it made for?

PANDORA Toolbox is customized for neuroscientific research. However, the concepts of a complex dataset, extraction of multiple observations from each item of the dataset, and analysis of multi-dimensional parameter spaces are universal. In its current form the database and dataset classes can be used for data other than electrophysiologic sources. As this toolbox is designed for flexible extensibility, one can add extensions that deal with different types of data and analyses.

1.6 Finding your way around

The source code uses MATLAB's documentation system, therefore all methods and classes are documented. To get help about all classes, issue the

```
>> help djlab
```

at the MATLAB prompt. This should give you an overview of available classes. Then, to learn about a specific class, ask for the documentation for the constructor method. For instance, for the `trace` class, issuing

```
>> help trace
```

gives you the documentation for the constructor together with an overview for the class. Sometimes, if there are multiple methods with the same name under different classes, you may get the wrong documentation. In that case, you can specify the class from which to take the method by prepending the class name to the method, such as in

```
>> help trace/spikes
```

In order to learn all methods available for a class, you can use MATLAB's `methods` command. For the `trace` class, do

```
>> methods(trace)
```

However, some documentation may be outdated or simply wrong. Please report these to the author via e-mail to `cgunay@emory.edu`.

1.7 Overview of this document

Next, Section 2 guides the reader through the installation of the package and other dependencies. You can skip this section if you already have a running software environment. Section 3 introduces the essential components of the software and talks about their design decisions. You can also skip this part if you're not interested in the guts of the system and you are in favor of a quick start. The recipes in Section 5 provide a tutorial for some common tasks. It may be easier for some readers to follow these recipes to jump-start using the software. However, it is recommended that you familiarize yourself with the basic organization of the classes before proceeding into more complex tasks. Section 6 takes the tutorial approach to describe common visualization tasks. Finally, Section A points to the list of individual methods provided by the software. These methods are documented in detail using the MATLAB online help system.

2 Installation

Download the latest package file from:
http://userwww.service.emory.edu/~cgunay/pandora.
Unpack the archive anywhere in your system, using

```
$ tar xzf pandora-xyz.tar.gz
```

and follow the instructions in the README file.

Basically it involves pointing your MATLAB installation to look at the pandora/ subdirectory for loading the PANDORA files. This can be achieved by adding this directory to your MATLAB search path using the addpath Matlab command. To avoid its repeated application for each new session, you can have a startup script, startup.m, in the directory that you run MATLAB with the following commands:

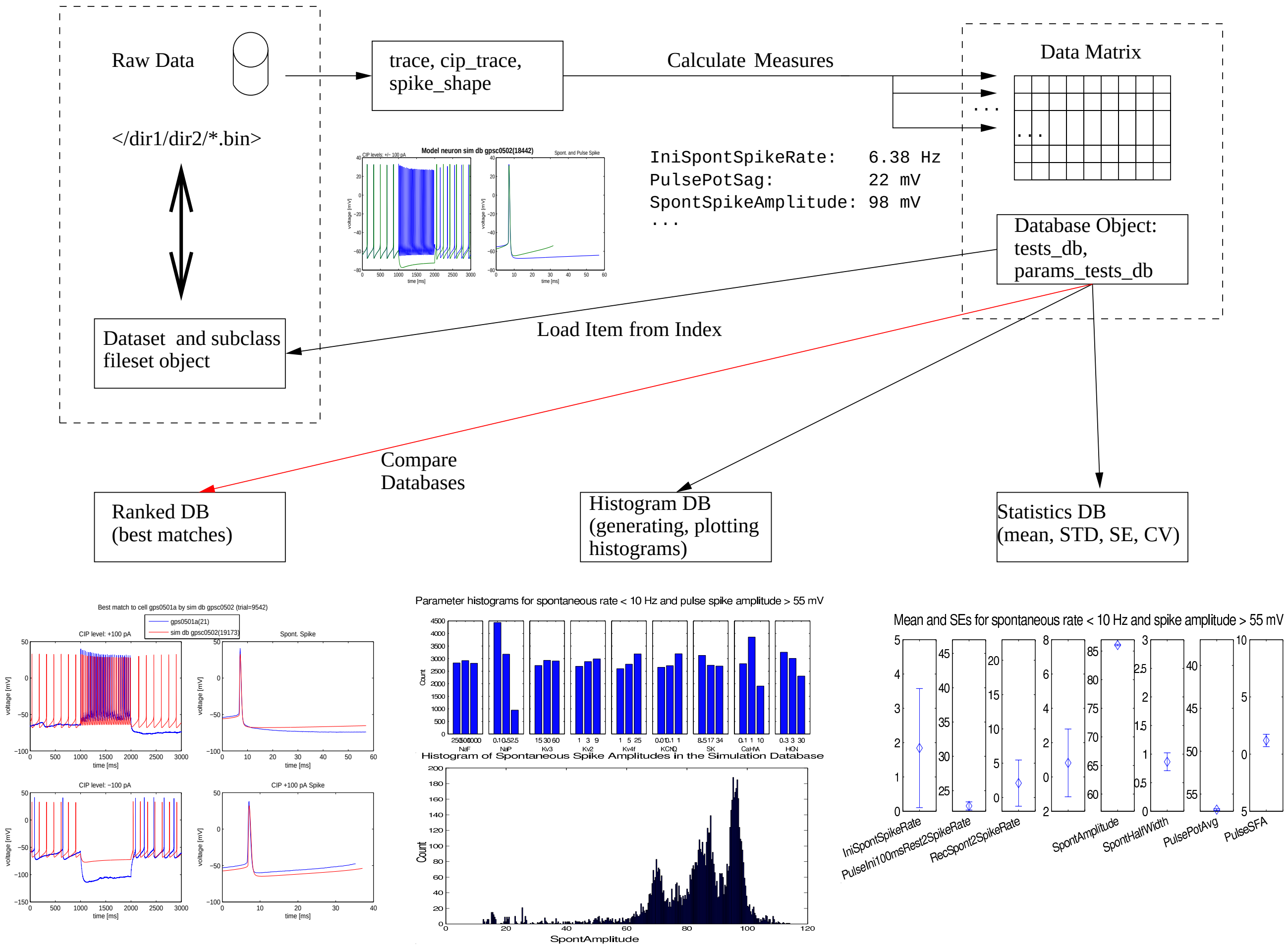
```
%--- startup.m for matlab
addpath /my/download/directory/pandora-1.0b/pandora
%--- end startup.m
```

This will be loaded everytime you run MATLAB from this directory. in UN*X systems, this can be improved further by placing the command in the file $HOME/matlab/startup.m, which is executed no matter from where MATLAB is called, especially if you are running Matlab from different or unknown places each time. In Windows, place the file under My Documents/MATLAB, or add the directory to the search path using the *File->Set Path* menu option.

Part I
Software Architecture

3 Toolbox Components

An overview of the toolbox functionality is shown in Figure 1. In the figure, boxes represent objects that can be created with the toolbox. Flow starts from the dataset object on the top left which represents the collection of raw data files. The raw data is loaded using information in the dataset to create intermediate objects that, for instance, contain data traces. These objects define electrophysiological measurements to be entered into the data matrix of the database object on the top right. The database object allows filtering and querying to refine its contents. From the database object, one can always go back to the dataset and find the raw data that results from a query. The arrows going to bottom objects and corresponding plots show the types of possible analyses that can be done on a database object. These analyses are typically for displaying statistical information. The red arrow is a speacial analysis for searching and matching rows between different databases. The match is done by taking a row from a database created with data from real neurons and finding best matching model neurons from a simulation database.



```
IniSpontSpikeRate:   6.38 Hz
PulsePotSag:         22 mV
SpontSpikeAmplitude: 98 mV
...
```

Raw Data
</dir1/dir2/*.bin>
Dataset and subclass fileset object
trace, cip_trace, spike_shape
Calculate Measures
Data Matrix
. . .
Database Object: tests_db, params_tests_db
Load Item from Index
Compare Databases
Ranked DB (best matches)
Histogram DB (generating, plotting histograms)
Statistics DB (mean, STD, SE, CV)
Model neuron sim db gpsc0502(18442)
Best match to cell gps0501a by sim db gpsc0502 (trial=9542)
Parameter histograms for spontaneous rate < 10 Hz and pulse spike amplitude > 55 mV
Histogram of Spontaneous Spike Amplitudes in the Simulation Database
SpontAmplitude
Count
Mean and SEs for spontaneous rate < 10 Hz and spike amplitude > 55 mV
IniSpontSpikeRate
PulseIni100msRest2SpikeRate
RecSpont2SpikeRate
SpontAmplitude
SpontHalfWidth
PulsePotAvg
PulseSFA

Figure 1: Schematic overview of the database software.

The objects in the figure are instances of classes that define their properties in the object-oriented framework. Each class comes with a hierarchy of of subclasses that specialize to specific functions. Subsequent sections describe each of these class hierarchies that make up the main components of the toolbox.

3.1 Databases hold all the information

The database object is at the center of this toolbox (see Figure 1). It holds a data matrix with rows as observations and columns as attributes. The rows would normally correspond to results from individual data traces, or simply neurons. The columns hold values of separate measurements, statistical data, or parameter values.

A database object can be created from any of the classes in the hierarchy of Figure 2. The top-level database class is `tests_db` which contains a two-dimensional data matrix of real numbers and some metadata. The metadata consists of column labels (e.g., measure names), a dataset label, and data properties (e.g., time resolution). The subclasses are specialized for different tasks.

If the database object is created using a dataset object, this maintains a connection from the elements of the database (e.g., neurons) to the raw data. This allows raw data associated with database contents to be visualized during analysis. However, a database can be created from any data matrix given in the proper format.

Some specialized subclasses of `tests_db` are as follows:

`params_tests_db` The first `num_params` columns are reserved for parameters that were changed between different rows. It contains methods that treat these columns specially. Parameters can be simulation parameters, or pharmacological applications to experiments.

`tests_3D_db` Contains a three-dimensional data matrix that has additional dimension for pages of information. This is mainly used to look at change in measurements with a parameter using the `invarParam` method of `params_tests_db`. Three dimensional databases can be useful for other purposes as well.

`stats_db` Contains few rows that describe the statistics obtained possibly from another database. It can contain the mean and standard deviation or error, or in some cases, the minimal and maximal values of columns in a database. It contains special plotting functions. There are methods that use the statistics collected by this class.

`ranked_db` Contains distances that resulted from a comparison of a database with a criterion. Its rows are ranked and sorted according to this distance value. Each row would point to a row in dex into the original database. Contains methods to generate reports from information about matching neurons.

`spikes_db` Contains results from individual spike shapes of a `trace` object. It can be obtained using the `trace/analyzeSpikesInPeriod` method.

`histogram_db` Each row corresponds to a histogram bin. Contains plotting methods.

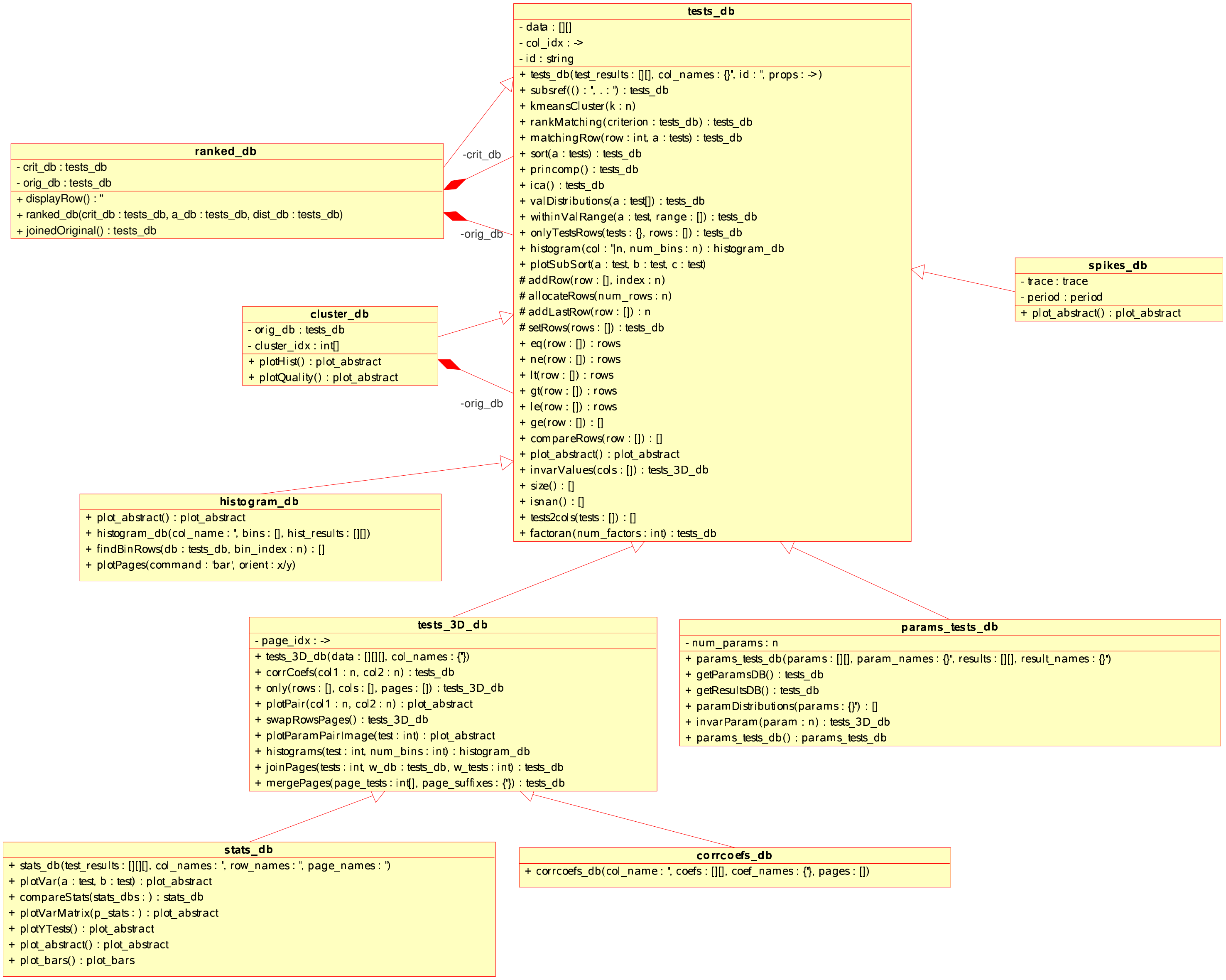

tests_db
- data : [][]
- col_idx : ->
- id : string
+ tests_db(test_results : [][], col_names : {}'', id : '', props : ->)
+ subsref(() : '', . : '') : tests_db
+ kmeansCluster(k : n)
+ rankMatching(criterion : tests_db) : tests_db
+ matchingRow(row : int, a : tests) : tests_db
+ sort(a : tests) : tests_db
+ princomp() : tests_db
+ ica() : tests_db
+ valDistributions(a : test[]) : tests_db
+ withinValRange(a : test, range : []) : tests_db
+ onlyTestsRows(tests : {}, rows : []) : tests_db
+ histogram(col : ''|n, num_bins : n) : histogram_db
+ plotSubSort(a : test, b : test, c : test)
# addRow(row : [], index : n)
# allocateRows(num_rows : n)
# addLastRow(row : []) : n
# setRows(rows : []) : tests_db
+ eq(row : []) : rows
+ ne(row : []) : rows
+ lt(row : []) : rows
+ gt(row : []) : rows
+ le(row : []) : rows
+ ge(row : []) : []
+ compareRows(row : []) : []
+ plot_abstract() : plot_abstract
+ invarValues(cols : []) : tests_3D_db
+ size() : []
+ isnan() : []
+ tests2cols(tests : []) : []
+ factoran(num_factors : int) : tests_db
ranked_db
- crit_db : tests_db
- orig_db : tests_db
+ displayRow() : ''
+ ranked_db(crit_db : tests_db, a_db : tests_db, dist_db : tests_db)
+ joinedOriginal() : tests_db
-crit_db
-orig_db
cluster_db
- orig_db : tests_db
- cluster_idx : int[]
+ plotHist() : plot_abstract
+ plotQuality() : plot_abstract
-orig_db
spikes_db
- trace : trace
- period : period
+ plot_abstract() : plot_abstract
histogram_db
+ plot_abstract() : plot_abstract
+ histogram_db(col_name : '', bins : [], hist_results : [][])
+ findBinRows(db : tests_db, bin_index : n) : []
+ plotPages(command : 'bar', orient : x/y)
tests_3D_db
- page_idx : ->
+ tests_3D_db(data : [][][], col_names : {''})
+ corrCoefs(col1 : n, col2 : n) : tests_db
+ only(rows : [], cols : [], pages : []) : tests_3D_db
+ plotPair(col1 : n, col2 : n) : plot_abstract
+ swapRowsPages() : tests_3D_db
+ plotParamPairImage(test : int) : plot_abstract
+ histograms(test : int, num_bins : int) : histogram_db
+ joinPages(tests : int, w_db : tests_db, w_tests : int) : tests_db
+ mergePages(page_tests : int[], page_suffixes : {''}) : tests_db
params_tests_db
- num_params : n
+ params_tests_db(params : [][], param_names : {}'', results : [][], result_names : {}'')
+ getParamsDB() : tests_db
+ getResultsDB() : tests_db
+ paramDistributions(params : {}'') : []
+ invarParam(param : n) : tests_3D_db
+ params_tests_db() : params_tests_db
stats_db
+ stats_db(test_results : [][][], col_names : '', row_names : '', page_names : '')
+ plotVar(a : test, b : test) : plot_abstract
+ compareStats(stats_dbs :) : stats_db
+ plotVarMatrix(p_stats :) : plot_abstract
+ plotYTests() : plot_abstract
+ plot_abstract() : plot_abstract
+ plot_bars() : plot_bars
corrcoefs_db
+ corrcoefs_db(col_name : '', coefs : [][], coef_names : {''}, pages : [])

Figure 2: Database class hierarchy.

`corrcoefs_db` Each row corresponds to a correlation coefficient. Contains plotting methods.

`cluster_db` Each row corresponds to a cluster centroid. Contains plotting methods.

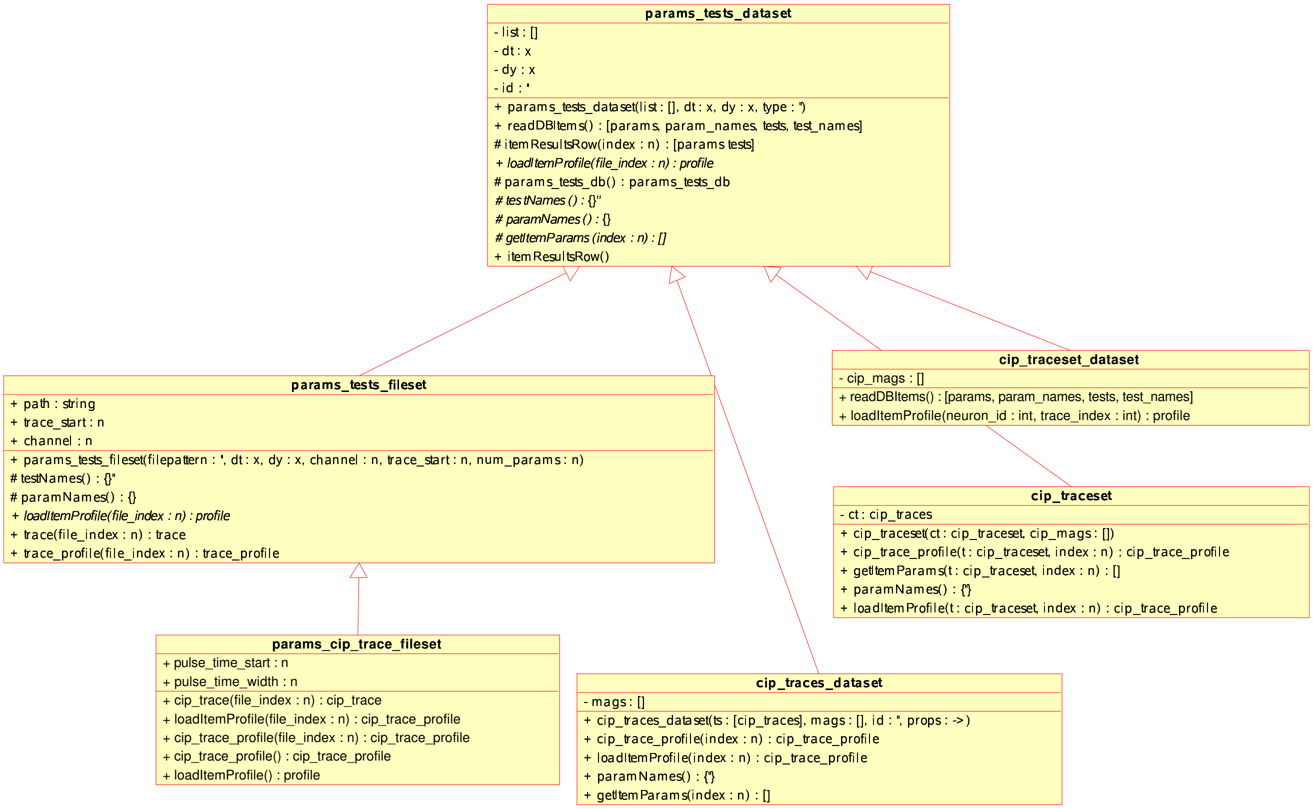

params_tests_dataset
- list : []
- dt : x
- dy : x
- id : '
+ params_tests_dataset(list : [], dt : x, dy : x, type : '')
+ readDBItems() : [params, param_names, tests, test_names]
# itemResultsRow(index : n) : [params tests]
+ loadItemProfile(file_index : n) : profile
# params_tests_db() : params_tests_db
# testNames () : {}''
# paramNames () : {}
# getItemParams (index : n) : []
+ itemResultsRow()
params_tests_fileset
+ path : string
+ trace_start : n
+ channel : n
+ params_tests_fileset(filepattern : ', dt : x, dy : x, channel : n, trace_start : n, num_params : n)
# testNames() : {}''
# paramNames() : {}
+ loadItemProfile(file_index : n) : profile
+ trace(file_index : n) : trace
+ trace_profile(file_index : n) : trace_profile
cip_traceset_dataset
- cip_mags : []
+ readDBItems() : [params, param_names, tests, test_names]
+ loadItemProfile(neuron_id : int, trace_index : int) : profile
cip_traceset
- ct : cip_traces
+ cip_traceset(ct : cip_traceset, cip_mags : [])
+ cip_trace_profile(t : cip_traceset, index : n) : cip_trace_profile
+ getItemParams(t : cip_traceset, index : n) : []
+ paramNames() : {''}
+ loadItemProfile(t : cip_traceset, index : n) : cip_trace_profile
params_cip_trace_fileset
+ pulse_time_start : n
+ pulse_time_width : n
+ cip_trace(file_index : n) : cip_trace
+ loadItemProfile(file_index : n) : cip_trace_profile
+ cip_trace_profile(file_index : n) : cip_trace_profile
+ cip_trace_profile() : cip_trace_profile
+ loadItemProfile() : profile
cip_traces_dataset
- mags : []
+ cip_traces_dataset(ts : [cip_traces], mags : [], id : '', props : ->)
+ cip_trace_profile(index : n) : cip_trace_profile
+ loadItemProfile(index : n) : cip_trace_profile
+ paramNames() : {''}
+ getItemParams(index : n) : []

Figure 3: Dataset class hierarchy.

3.2 Datasets create the databases

The dataset object is responsible for creating the database objects (see Figure 1). It defines where the raw data is stored and what parameters are used to load and analyze it. It knows that raw data has parameters associated which individual raw data traces and how and which measures will be generated. This information is used to automatically generate a database from the dataset. It also allows reaching back the raw data from rows of an analyzed database.

Figure 3 shows the hierarchy for the dataset classes. The top-level dataset class is `params_tests_dataset` which is an incomplete class. That is, this class defines general utilities that can work for a variety of dataset subclasses, but one cannot make a object from the `params_tests_dataset` class directly. Instead, one of its subclasses must be chosen and used. Some of these specialized subclasses are as follows:

`params_tests_fileset` This class assumes each raw data item resides in a file and all of these files are in the same directory. The parameter names and values are obtained from each file name itself. This class is mostly useful for simulation filesets.

`params_cip_trace_fileset` This is a subclass of `params_tests_fileset`, therefore it inherits the notion of one file per data item. The files must conform to the current-pulse injection experiments and have a starting time and duration for the pulses. The pulse magnitude is read from the `pAcip` parameter. This class is mostly useful for simulation filesets.

`physiol_cip_traceset` This is a subclass of `params_tests_dataset`. It is designed to load a set of physiology traces from a single file generated by the PCDX stimulation and acquisition software.

`physiol_cip_traceset_fileset` This is a subclass of `params_tests_dataset`. It is designed to load traces from multiple PCDX data files. It uses the `physiol_cip_traceset` class for this purpose.

`cip_traces_dataset, cip_traceset, cip_traceset_dataset` These are obsolete classes that allow loading physiology traces from older MATLAB formatted objects.

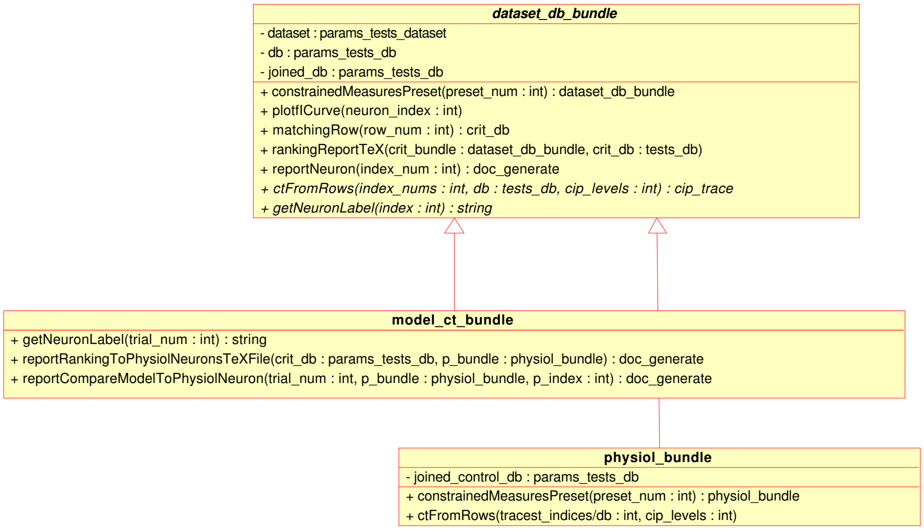

dataset_db_bundle
- dataset : params_tests_dataset
- db : params_tests_db
- joined_db : params_tests_db
+ constrainedMeasuresPreset(preset_num : int) : dataset_db_bundle
+ plotfICurve(neuron_index : int)
+ matchingRow(row_num : int) : crit_db
+ rankingReportTeX(crit_bundle : dataset_db_bundle, crit_db : tests_db)
+ reportNeuron(index_num : int) : doc_generate
+ ctFromRows(index_nums : int, db : tests_db, cip_levels : int) : cip_trace
+ getNeuronLabel(index : int) : string
model_ct_bundle
+ getNeuronLabel(trial_num : int) : string
+ reportRankingToPhysiolNeuronsTeXFile(crit_db : params_tests_db, p_bundle : physiol_bundle) : doc_generate
+ reportCompareModelToPhysiolNeuron(trial_num : int, p_bundle : physiol_bundle, p_index : int) : doc_generate
physiol_bundle
- joined_control_db : params_tests_db
+ constrainedMeasuresPreset(preset_num : int) : physiol_bundle
+ ctFromRows(tracest_indices/db : int, cip_levels : int)

Figure 4: Bundle class hierarchy.

3.3 Bundling the database and dataset together

Since dataset and database objects are related and work together for some operations, it is convenient to have another object that bundles them together. There are several analysis routines that start from the database, retrieve raw data traces and other related information from the dataset and create a result. For instance, matching neurons from one database to another requires first comparing the measurements to find match candidates, and then comparing raw traces to visually represent the match quality.

The top-level `dataset_db_bundle` class in Figure 4 fulfills this purpose by bundling a `dataset` with the raw database, `db`, created from it, and with the reduced database, `joined_db`, that contains a one-row-per-neuron representation. Although being a virtual class that cannot be instantiated, it contains general methods and prototype methods that must be implemented in subclasses. This way, it provides guidelines for defining subclasses. Its two subclasses provide specialize methods for model and physiology databases, respectively.

`model_ct_bundle` Contains methods to name and visualize neurons in the model database. It has methods to compare real neurons to model neurons to find best matching candidates.

`physiol_bundle` Contains methods to name and visualize neurons in the physiology database. It contains a new attribute, `joined_control_db`, that holds only the neurons recorded without any pharmacological treatments.

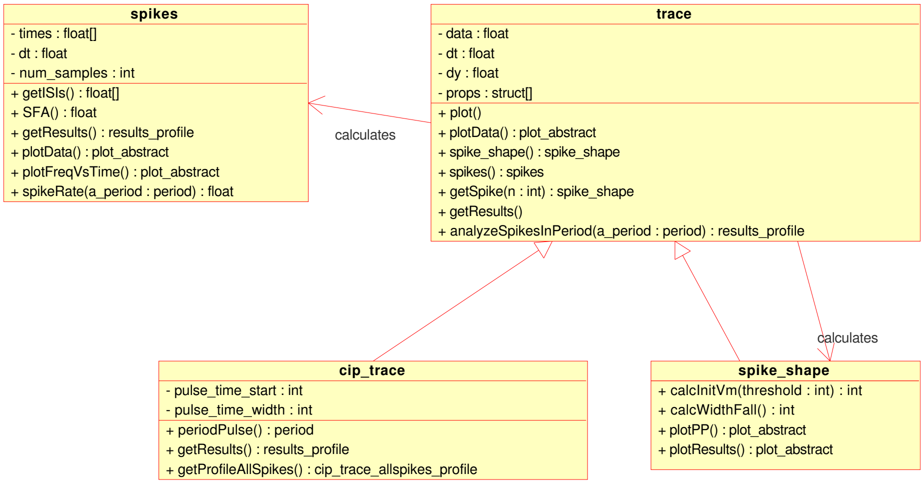

spikes
- times : float[]
- dt : float
- num_samples : int
+ getISIs() : float[]
+ SFA() : float
+ getResults() : results_profile
+ plotData() : plot_abstract
+ plotFreqVsTime() : plot_abstract
+ spikeRate(a_period : period) : float
calculates
trace
- data : float
- dt : float
- dy : float
- props : struct[]
+ plot()
+ plotData() : plot_abstract
+ spike_shape() : spike_shape
+ spikes() : spikes
+ getSpike(n : int) : spike_shape
+ getResults()
+ analyzeSpikesInPeriod(a_period : period) : results_profile
calculates
cip_trace
- pulse_time_start : int
- pulse_time_width : int
+ periodPulse() : period
+ getResults() : results_profile
+ getProfileAllSpikes() : cip_trace_allspikes_profile
spike_shape
+ calcInitVm(threshold : int) : int
+ calcWidthFall() : int
+ plotPP() : plot_abstract
+ plotResults() : plot_abstract

Figure 5: Data wrapper class hierarchy.

3.4 Wrapper classes hold raw data

Wrapper classes are designed to hold data and provide simple methods that operate on them. They can either hold raw data, or intermediate processed forms of data being byproducts of analysis routines. In the overall schema of Figure 1, the raw traces obtained from the dataset object are kept in data wrapper objects.

Figure 5 shows the hierarchy for the data wrapper classes. The most basic data wrapper class in this toolbox is the `trace` class, which holds raw voltage or current traces. The `spikes` object contains the spike times obtained by analyzing a `trace` object.

A data wrapper class does more than just holding the data. It defines a set of operations in terms of method functions that can work on the data held by the class. As a rule of thumb, if one needs to add some new functionality into the toolbox, it should be added as a method into a class holding the data on which to operate.

Some of the data wrapper classes are as follows:

`trace` Generic object that holds a vector of data that changes over time. It has a time resolution and y-axis resolution. Contains simple analysis routines such as finding average values within different periods, or finding spikes given a threshold.

`cip_trace` A subclass of `trace` class for current-injection recording protocols. It defines an initial spontaneous period, followed by a current-injection period, and final recovery period. It contains period-specific analyses that apply to the experimental protocol.

`spike_shape` A subclass of `trace` that holds the shape of a single spike. It contains spike shape measurements.

`spikes` A generic class to hold the event times for spikes. It contains methods for making measurements based on spike times, such as rate and ISI calculations.

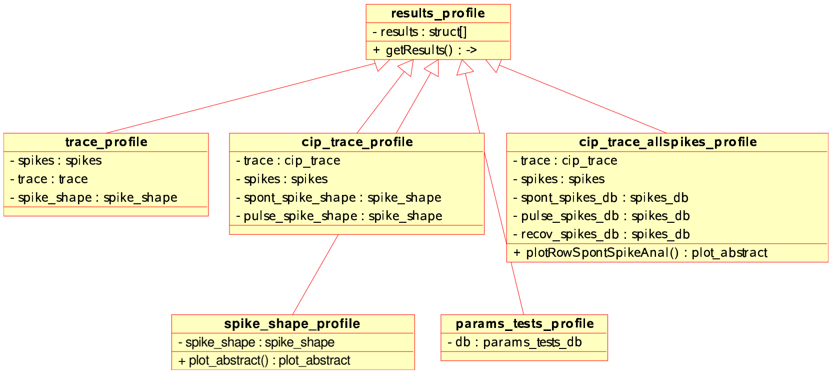

results_profile
- results : struct[]
+ getResults() : ->
trace_profile
- spikes : spikes
- trace : trace
- spike_shape : spike_shape
cip_trace_profile
- trace : cip_trace
- spikes : spikes
- spont_spike_shape : spike_shape
- pulse_spike_shape : spike_shape
cip_trace_allspikes_profile
- trace : cip_trace
- spikes : spikes
- spont_spikes_db : spikes_db
- pulse_spikes_db : spikes_db
- recov_spikes_db : spikes_db
+ plotRowSpontSpikeAnal() : plot_abstract
spike_shape_profile
- spike_shape : spike_shape
+ plot_abstract() : plot_abstract
params_tests_profile
- db : params_tests_db

Figure 6: Profile class hierarchy.

3.5 Profiles hold results of measurements

Profile classes are designed to hold results of analysis and measurements on the data wrapper or database objects. The data and results are separated into different classes for added flexibility of saving data and results separately. Yet, the profiles normally keep a copy of the data wrapper object from which they obtained the measurements. The intention is to save the measurement results for possible visualization or later inspection, without having to repeat the analyses.

In Figure 6, the top-level `results_profile` class contains a simple MATLAB structure variable, `results`, that holds a set of name-value pairs. These are names of measurements and their corresponding values. Most of the subclasses are simplistic, and they exist only for organizational reasons. Some of them may implement specialized plotting methods that make use of the saved measurements. These subclasses can be briefly described as follows:

`trace_profile` Holds measurements from a `trace` object. It contains the `trace` object and the `spikes` found in it, and averaged `spike_shape` object.

`cip_trace_profile` Holds measurements from a `cip_trace` object with a current-injection period. It contains the original `cip_trace` object and the `spikes` found in it. In addition, it holds averaged `spike_shape` objects from the spontaneous and current-injection periods.

`cip_trace_allspikes_profile` Extended version of `cip_trace_profile`. Instead of single averaged spike shapes, it contains spike databases from the spontaneous, current-injection and recovery periods. These databases only retain measurements made from individual spikes, but not their shapes.

`spike_shape_profile` Holds measurements made from a `spike_shape` object.

`params_tests_profile` Holds analysis results from a `params_tests_db` object.

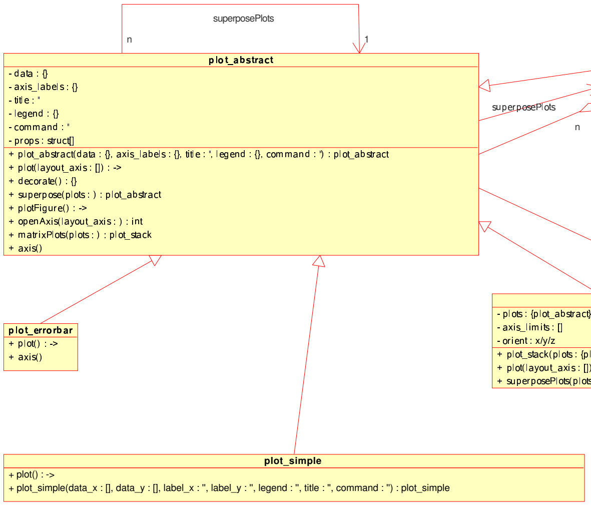

superposePlots
n
1
plot_abstract
- data : {}
- axis_labels : {}
- title : ''
- legend : {}
- command : ''
- props : struct[]
+ plot_abstract(data : {}, axis_labels : {}, title : '', legend : {}, command : '') : plot_abstract
+ plot(layout_axis : []) : ->
+ decorate() : {}
+ superpose(plots :) : plot_abstract
+ plotFigure() : ->
+ openAxis(layout_axis :) : int
+ matrixPlots(plots :) : plot_stack
+ axis()
superposePlots
1
n
plot_superpose
- plots : plot_abstract{}
+ decorate()
+ axis()
+ superposePlots(plots : plot_superpose) : plot_superpose
superposePlots
n
superposePlots
1
n
n
plot_stack
- plots : {plot_abstract}
- axis_limits : []
- orient : x/y/z
+ plot_stack(plots : {plot_abstract}, axis_limits : [], horiz_vert : 0/1) : plot_stack
+ plot(layout_axis : []) : ->
+ superposePlots(plots :) : plot_stack
plot_errorbar
+ plot() : ->
+ axis()
plot_simple
+ plot() : ->
+ plot_simple(data_x : [], data_y : [], label_x : '', label_y : '', legend : '', title : '', command : '') : plot_simple
plot_bars
plot_errorbars
+ plot()

Figure 7: Plot classes hierarchy.

3.6 Integrated plotting for easy visualization

To integrate visualization into each class, common MATLAB plotting features are implemented in the supporting classes seen in Figure 7. These bring an object-oriented approach to plot generation in MATLAB. Plots can be generated as objects, saved, modified and included as subplots in larger plots.

The main plotting classes are plot_abstract, plot_superpose, and plot_stack. The most general plotting template class, and the top-level class in the hierarchy, is plot_abstract, which plots an axis using a single MATLAB command, like plot or bar. Multiple plot_abstract objects that use the same command can be superposed and still act as a single plot_abstract object. If they require different plotting commands (e.g., mixing plot and text labels), a plot_superpose object must be used that is composed of an array of plot_abstract objects. Multiple plot_abstract objects or any of the subclass objects can be composed together in a horizontal or vertical stack using the plot_stack class. Since plot_stack is itself a subclass of plot_abstract, it can be stacked as well. This allows creating virtually any complex structured figure using the three classes. Each of these classes have several properties that control the layout and details of placement and looks.

The rest of the classes in the hierarchy create typical types of plots for convenience:

plot_bars Multi-axis bar plot with extending errorbars using a combination of the bar, errorbar, and text commands.

plot_errorbar Single-axis errorbar plot using the errorbar command.

plot_errorbars Multi-axis errorbar plot using the errorbar command.

plot_simple Simplified single-axis, single command plot.

3.7 Miscellaneous classes

These are miscellaneous classes that do not fit into any of the above categories:

`period` Defines a period composed of a start and end time for operations on traces, etc.

`script_array` Defines a looping construct that can be extended. It defines an initialization routine, a job that needs to be repeated, and a finalization routine.

`script_array_for_cluster` Subclass of `script_array`, it can submit the array job to run in parallel on a computing cluster that supports the Sun Grid Engine (SGE) commands.

`script_factory` Factory class to generate an enumerated array of script files to be distributed on several machines and run in parallel. It also defines a final function to gather results. It is recommended to use `script_array_for_cluster` instead.

4 Programming Conventions

4.1 Using property structures for passing optional arguments to methods

For flexibility in passing optional arguments to methods, this toolbox adopted using property structures. A MATLAB structure, mostly called `props`, is passed to a method as the last argument:

```
>> props.optionalParam = 1
>> myFunc('hello', props)
```

Each method defines a list of accepted arguments that can be defined as fields in the structure, but should be able to execute without them by substituting defaults. Using a property structure is advantageous over using the `varargin` keyword for variable number of arguments, because properties allow adding and deleting arguments in methods without causing changes to the contents of the method. Since arguments are addressed by names rather than positional arguments, missing arguments do not affect the other arguments.

Most objects keep a property structure that define custom attributes passed at time of construction. These classes define a `setProp` method to modify properties after being created.

4.2 Overloaded operators for transparent access to object contents

The simplistic implementation of object-oriented programming features in Matlab impose several strict limitations. MATLAB's powerful and flexible operator overloading feature helps overcome these limitations.

PANDORA Toolbox uses MATLAB operator overloading to facilitate manipulation of local and parent object fields. In MATLAB, object fields can only be accessed from the object's private methods. This means one cannot access the object fields using the dot operator. To give an example, the `trace` object has a `dt` field for time resolution. The following command fails:

```
>> mytrace.dt = 1e-4;
??? Object fields can only be accessed within methods.
```

Everytime object contents need to be addressed, a method must be called. The recommended way to do this is by defining separate getter/setter methods for each field of the object. For instance, writing `getDt` and `setDt` methods for accessing the `dt` field. This creates a lot of burden for the programmer not just creating a class, but also maintaining it later. Although this probably was intended for strictness in building object-oriented constructs, it is highly inconvenient for command-line manipulations. Therefore our toolbox objects offer generic `get` and `set` methods that can read or write the value of any of its fields:

```
>> mytrace = set(mytrace, 'dt', 1e-4)
>> get(mytrace, 'dt')
ans = 1e-04
```

These methods are almost identical across different classes. In addition to this, defining the special `subsref` method for objects allow overloading the dot (`.`), parenthesis (`()`), and curly brace (`{}`) operators. Most[3] of the objects in the toolbox allows using the dot operator to read or write to fields. Overloading these operators also help with the limitation of accessing parent object fields, a problem not found in other object-oriented languages such as JAVA. For example without any overloading, from the subclass class `cip_trace` one needs to first address the parent class name, and then `dt`:

```
>> myciptrace.trace.dt
ans = 1e-4
```

After defining the overloaded operator that call parent methods, one get reach `dt` directly:

```
>> myciptrace.dt
ans = 1e-4
```

Some classes overload indexing operators to allow accessing special functions. For instance the main database class, `tests_db`, overloads parenthesized indexing to access cells in the database matrix. Some classes define the special `subsasgn` method to overload the assignment operations when the object is on the left-hand-side of the operation. This allows the command:

```
>> mytrace.dt = 1e-4;
```

which would otherwise need to be done the following way:

```
>> mytrace = set(mytrace, 'dt', 1e-4);
```

[3] May not be implemented for all objects.

4.3 Troubleshooting errors

For debugging problems with methods, one can turn on the verbosity of information display during execution with:

```
>> warning on verbose
>> warning on backtrace
```

4.4 Creating a new class

To get the benefit of overloading, the top-level class must have the generic `subsref` and `subsasgn` methods. These methods can be copied from any of the other top-level classes. Any subclasses should have the generic `get` and `set` methods in place.

Part II
User's Manual

5 Recipes for Common Tasks

5.1 Loading a database

A database can be created directly from a data matrix, or indirectly by loading a dataset. For the latter, first a dataset object must be created that point to the data sources. There are different dataset classes that allow using different data sources. For instance, physiology and simulation data require different operations. In physiology data, one can record information about the treatments and other conditions, whereas in simulations one can keep track of changing parameters.

Once the dataset object is obtained, the database object can be created with

```
>> mydb = params_tests_db(dataset)
```

which initiates the loading of files. This operation is the same no matter what type of dataset or fileset object is used. The following commands reduce the verbosity of output during this long process:

```
>> warning off verbose
>> warning off backtrace
>> warning off calcInitVm:info
```

5.1.1 Creating a dataset for physiology data

Physiology data can be obtained from multiple sources.

Loading data by specifying tracesets in a text file The preferred way to load physiology traces is to first create a text file where each line specifies traces to load from a single data source (e.g., a PCDX file). The format of this text file is explained in the help of the physiol_cip_traceset_fileset class. The physiology fileset can be created from the text file with a command such as follows:

```
>> phys_fileset =
     physiol_cip_traceset_fileset('cell_traces.txt', 1e-4, 1e-3,
                                  struct('profile_class_name',
                                         'getProfileAllSpikes',
                                         'offset_y', -9,
                                         'cip_list',
                                         [-200 -100:20:100 200 300]))
```

This command reads the cell_traces.txt file and records the tracesets to read from each file. The structure passed indicates to use the getProfileAllSpikes method to calculate the measurements on the traces.

The fileset can then be used to generate the database, as shown above, using its params_tests_db method. The fileset object holds within, a separate physiol_cip_traceset object for each line in the text file.

The cip_list optional parameter must be used with caution. To determine actual CIP-levels, the current channel of the trace is analyzed. cip_list entries are used to discretize the noisy current channel. Current levels will coerce to the nearest entry from cip_list. In the above example, all current levels below -200 pA will be assumed to be -200 pA. The default cip_list resides in the physiol_cip_traceset/CIPform method.

Loading data from existing cip_traces objects The now obsolete cip_traces Matlab objects have been used to hold some earlier physiological data. Each object holds a set of traces with varying CIP levels applied to the cell. The following command creates a dataset object from a cell array ct_list of cip_traces objects by choosing only the traces with 100 pA and -100 pA CIP levels

```
>> phys_dataset =
   cip_traceset_dataset(ct_list, [100, -100], 1e-3,
                        'dataset gpd 0411-21',
                        struct('offsetPotential', -9)
```

5.1.2 Creating a dataset for simulation data

First a dataset or a fileset must be created. An example to load GENESIS .bin files would be

```
>> fileset =
   params_cip_trace_fileset('/home/cengiz/data/*.bin',
                            1e-4, 1e-3,
                            20001, 10000,
                            'sim dataset gpsc0501',
                            struct('trace_time_start', 10001,
                                   'type', 'sim',
                                   'scale_y', 1e3))
```

The explanation of arguments can be obtained by issuing a

```
>> help params_cip_trace_fileset
```

in MATLAB. In this example, all GENESIS files were created with the same characteristics: $\mathrm{d}t = 10^{-4}$, $\mathrm{d}V = 10^{-3}$, pulse during samples $[20001, 30000]$. Optional properties (the last argument) indicates that the first 10000 samples should be discarded and that the data should be prescaled to yield the $\mathrm{d}V$ indicated. Note that, using an absolute path to refer to data files ensures that they can be reached from different directories after the fileset object is saved.

Loading heterogeneous set of simulation files Sometimes not all data files in a simulation set would have the same length, or CIP start time. The brute-force simulation set is such an example, where the spontaneous trace and different CIP level traces are in different files. I have a special superclass that contains multiple fileset objects to automatically handle this kind of data. It resides not in the general distribution directory, but in my personal directory `/djlab/shared/matlab/classes/cengiz`. This class is an example of how to create composite fileset objects. An instance of this class can be created with:

```
>> m_filesetall =
     multi_fileset_gpsim_cns2005('.../data', '.../paramRanges.txt',
                                  '.../all_0.par', 'sim db gpsc0502')
```

which will find all the files in the given directory and put them in separate pre-defined fileset objects according to their `_pAcip_` suffixes. Parameter range definition and value files are used to read parameter values for each simulation. A single database object can be loaded using the `params_tests_db` method on the `m_filesetall` object above. The help on this method explains how to load only certain filesets at a time. This helps to load different filesets in parallel, since databases can be concatanated easily afterwards.

5.2 Finding constrained subsets in a database

Once a database with more-than-sufficient number of measures is available, subsets of this database can be extracted easily for other tasks. New databases can be formed by filtering rows, columns or pages of an existing database. For choosing any of these dimensions, the user can specify an array of indices, or a logical array. For instance,

```
>> db2 = db1(1:10, :);
```

creates a database object `db2` by the first ten rows of `db1` and all its columns. For three-dimensional databases, a third parameter can be specified, as in

```
>> db2 = db1(1:10, :, [1 3]);
```

which will take only the first and third page from `db1`.

For measures, columns can also be specified as a single string value, or a cell array of strings, as in

```
>> db2 = db1(1:10, 'pAcip');
```

which chooses only the `pAcip` column of the first 10 rows of `db1` or

```
>> db2 = db1(1:10, {'pAcip', 'IniSpontSpikeRate'});
```

which chooses two columns. Finally, composite queries can be formed when cell arrays are used for addressing:

```
>> db2 = db1(1:10, {1:10, 'IniSpontSpikeRate', 234});
```

which will select the first ten measures, the spontaneous spike rate, and the measure number 234.

Rows of the database signify neurons or simulation runs. Therefore it is important to find subset of neurons that match a certain criteria. This can be done specifying a list of rows that is the result of a logical operation. A logical operation finds rows that satisfy constraints on a single parameter or measure of a database. For instance,

```
>> rows = db1(:, 'IniSpontSpikeRate') > 10;
```

gives a logical array that contains a `true` value for all rows in `db1` that has spontaneous firing faster than 10 Hz. If this is used as the row specifier in a subset operation, a new database with only these neurons can be obtained by

```
>> db2 = db1(rows, :);
```

If we want more constrains it is straightforward to use any logical operation such as AND (&), OR (|) and NOT(~) on these logical arrays such as

```
>> db2 = db1((rows | rows2) & rows3, :);
```

which says choose all rows from `db1` where either the tests `rows` and `rows2` are satisfied and while `rows3` is always satisfied. All these operations can be specified in-place such as in

```
>> db2 = db1(db1(:, 'IniSpontSpikeRate') > 10 &
           db1(:, 'IniSpontSpikeRate') <= 20, :);
```

which will create a database of neurons that spontaneously fire between 10–20 Hz.

5.2.1 Complex Queries

Complex queries can be constructed using results of queries in nested fashion. The following example shows an example of finding all neurons that match any of the neurons in another database and then find the ones that match certain criteria:

```
>> sub_phys_es2 = phys_joined_db(phys_joined_db(:, 'NeuronId') == es2(:, 'NeuronId'), :);
>> displayRows(sub_phys_es2(sub_phys_es2(:, 'Apamin') > 0, 'NeuronId'))
```

Here, the first query returns all rows that match the `NeuronId`s from the `es3` database. The inner term in the second query finds among these neurons the ones for which apamin blocker data is present. The final enclosing block uses these rows to get a subset of the `phys_joined_db` with only the `NeuronId` column. This type query has equivalent computational power to using nested SELECT statements in the SQL language.

5.3 Sorting the database according to a measure

First, all rows where the desired measure value is NaN should be eliminated:

```
>> ampDecayTau_nonNaN_db =
     dball(~isnan(dball(:, 'PulseSpikeAmpDecayTau')), :)
```

This finds all rows in dball that the PulseSpikeAmpDecayTau measure is not NaN and creates a new DB object ampDecayTau_nonNaN_db, which includes these rows with all measures from dball. Notice that the newly created DB contains less rows than the original DB. The number of rows in the new DB can be obtained by just typing the name of the DB and pressing enter at the MATLAB prompt.

Then, one can sort the new database using:

```
>> ampDecayTau_sorted_db = sortrows(ampDecayTau_nonNaN_db,
                                    'PulseSpikeAmpDecayTau')
```

This will create DB which is sorted with increasing values of the PulseSpikeAmpDecayTau measure. Displaying the first few rows gives lowest values:

```
>> displayRows(ampDecayTau_sorted_db, 1:3)
ans =
    'NaF'                    [      1000]    [      250]    [      250]
    'NaP'                    [    0.5000]    [   0.5000]    [   2.5000]
    'Kv3'                    [        60]    [       15]    [       30]
    'Kv2'                    [         9]    [        3]    [        3]
    'Kv4f'                   [         5]    [        1]    [       25]
    'KCNQ'                   [    0.1000]    [   0.0100]    [   0.1000]
    'SK'                     [    8.5000]    [       34]    [   8.5000]
    'CaHVA'                  [        10]    [   0.1000]    [       10]
    'HCN'                    [        30]    [   0.3000]    [       30]
    'pAcip'                  [       100]    [      100]    [      100]
    'IniSpontISICV'          [3.9448e-04]    [   0.0051]    [   0.0452]
    'IniSpontPotAvg'         [  -64.9161]    [-52.6859]    [-41.5876]
    'IniSpontSpikeRate'      [   14.0014]    [ 18.0018]    [ 81.0081]
    'PulseISICV'             [    0.0226]    [        0]    [   0.0366]
    'PulseIni100msISICV'     [    0.0541]    [        0]    [   0.0814]
              [1x27 char]    [   28.8953]    [        0]    [ 88.9086]
              [1x27 char]    [   26.6785]    [        0]    [ 86.7052]
              [1x22 char]    [        30]    [       20]    [      100]
              [1x25 char]    [   29.4118]    [166.6667]    [ 96.0512]
    'PulsePotAvg'            [  -61.0044]    [-32.7775]    [-34.1518]
    'PulsePotMin'            [       NaN]    [      NaN]    [      NaN]
    'PulsePotSag'            [       NaN]    [      NaN]    [      NaN]
    'PulseSFA'               [    1.1254]    [      NaN]    [   1.3571]
    'PulseSpikeAmpDecayDelta'[    4.2764]    [   9.2041]    [ 17.8389]
    'PulseSpikeAmpDecayTau'  [    0.2000]    [   0.2000]    [   0.3000]
    'PulseSpikeRate'         [   28.0028]    [   2.0002]    [ 89.0089]
...
```

Displaying the last few rows gives the highest values:

```
>> displayRows(ampDecayTau_sorted_db, 13879:13881)
ans =
    'NaF'                     [      250]    [      250]    [       1000]
    'NaP'                     [  2.5000]    [  2.5000]    [     2.5000]
    'Kv3'                     [       15]    [       15]    [         60]
    'Kv2'                     [        3]    [        3]    [          9]
    'Kv4f'                    [        5]    [        5]    [         25]
    'KCNQ'                    [  0.1000]    [  0.1000]    [     0.0100]
    'SK'                      [  8.5000]    [  8.5000]    [         17]
    'CaHVA'                   [       10]    [       10]    [         10]
    'HCN'                     [       30]    [        3]    [         30]
    'pAcip'                   [     -100]    [     -100]    [        100]
    'IniSpontISICV'           [  0.0027]    [  0.0027]    [9.1376e-04]
    'IniSpontPotAvg'          [-28.5685]    [-28.5687]    [   -67.7567]
    'IniSpontSpikeRate'       [ 69.0069]    [ 69.0069]    [    14.0014]
    'PulseISICV'              [  0.0046]    [  0.0046]    [     0.0091]
    'PulseIni100msISICV'      [  0.0080]    [  0.0080]    [          0]
              [1x27 char]     [ 71.1269]    [ 71.1269]    [    24.4499]
              [1x27 char]     [ 71.1427]    [ 71.1427]    [    26.6785]
              [1x22 char]     [       80]    [       80]    [         20]
              [1x25 char]     [ 70.6357]    [ 70.6357]    [    25.4453]
    'PulsePotAvg'             [-30.1705]    [-30.1718]    [   -65.2721]
    'PulsePotMin'             [      NaN]    [      NaN]    [        NaN]
    'PulsePotSag'             [      NaN]    [      NaN]    [        NaN]
    'PulseSFA'                [  0.9792]    [  0.9792]    [     1.0407]
    'PulseSpikeAmpDecayDelta'[ -1.2791]    [ -1.2868]    [     1.2201]
    'PulseSpikeAmpDecayTau'   [999.6000]    [999.6000]    [       1000]
    'PulseSpikeRate'          [ 72.0072]    [ 72.0072]    [    25.0025]
...
```

5.4 Preprocessing a raw (physiology) database by elimination and averaging

Mostly, raw physiology databases are subject to redundancies and unwanted recordings. We usually apply the following steps before we start analyzing a raw physiology database. Similar steps may be employed for simulation databases, too.

5.4.1 Limiting range of bias currents

Recordings with high bias current are undesirable. We commonly filter-out high bias currents with:

```
>> db_bias_small =
     phys_dball(phys_dball(:, 'pAbias') > -30 &
                phys_dball(:, 'pAbias') < 30, :)
```

which will limit the bias current, i_b, to $-30\,\text{pA} < i_b < 30\,\text{pA}$.

5.4.2 Choosing few current levels

To get a profile for a neuron, usually both hyperpolarizing and depolarizing CIP-levels need to be included. Moreover, to capture the spiking frequency vs. current response of the neuron, multiple depolarizing CIP-level information may need to be included.

There are two counterparts to selecting which CIP-levels to include in a DB. First, one can select what CIP-levels are available in the raw data and what discretization levels should be used while loading the database. This is done with the `cip_list` optional parameter described in Section 5.1.1. GP recordings prior to mid-2005 have current channel data which are too noisy to be quantized to levels of 10 pA. Instead, at least a step size of 20 pA needs to be used. Later recordings have both better recordings, and feature 20 pA steps in the experimental protocol anyway.

Second, after the database is loaded, one can filter-out unwanted CIP-level traces:

```
>> phys_dball_limitedcip =
     phys_dball_big(phys_dball_big(:, 'pAcip') == -100 |
                    phys_dball_big(:, 'pAcip') == 0 |
                    phys_dball_big(:, 'pAcip') == 50 |
                    phys_dball_big(:, 'pAcip') == 100 |
                    phys_dball_big(:, 'pAcip') == 200, :)
```

This operation can be simplified to take advantage of complex query form:

```
>> phys_dball_limitedcip =
     phys_dball_big(phys_dball_big(:, 'pAcip') == [-100; 0; 50; 100; 200])
```

which will choose rows with current levels matching any of the given values.

5.4.3 Adding new columns calculated from existing measures

Some measures can be deduced from measures collected from raw data. These do not need to be calculated at time of loading the raw data, but rather can be added to the database later. Some measures must be added later because they may be composed of measurements from multiple traces or averages. Here is an example for adding a new measure:

```
>> phys_db_limitedcip_addedcols =
     addColumn(phys_db_limitedcip, 'PulsePotSagDivMin',
               phys_db_limitedcip(:, 'PulsePotSag').data ./
               phys_db_limitedcip(:, 'PulsePotMin').data)
```

5.4.4 Averaging multiple traces of same neuron with same CIP-level and (pharmacological) parameters

In making a one-row-per-CIP-level database, it is essential to include all available information from the raw database. Especially in physiology datasets, there may be multiple traces of a neuron where the same CIP-level and the same pharmacological conditions were applied. These rows can be averaged to obtain a single row for each CIP-level of a neuron.

Before doing this, the parameters of the raw database should only include parameters that uniquely distinguish neurons. The averaging operation tries to find each distinct set of parameters and then averages all rows that has this combination. For example, the `NeuronId` and `pAcip` parameters need to be distinct. However, the `pAbias` parameter does not need to be distinct for each neuron. The non-unique parameters need to be filtered-out before the averaging process. The following shows all the parameters of a raw physiology database:

```
>> phys_db
params_tests_db, tracesets from .../cip_traces_all_axoclamp.txt
ans =
    num_params: 16
         props: [0x0 struct]
      tests_db: [1x1 tests_db]
Optional properties of params_tests_db:
ans =
0x0 struct array with no fields.
tests_db, tracesets from .../cip_traces_all_axoclamp.txt
1527 rows in database with 182 columns, and 1 pages.
Column names:
    [  1]    ’pulseOn’
    [  2]    ’pulseOff’
    [  3]    ’traceEnd’
    [  4]    ’pAcip’
    [  5]    ’pAbias’
    [  6]    ’Cadmium’
    [  7]    ’PicroTx’
    [  8]    ’Apamin’
    [  9]    ’Glycine’
    [ 10]    ’KynAcid’
    [ 11]    ’TTX’
    [ 12]    ’XE991’
    [ 13]    ’drug_4AP’
    [ 14]    ’EBIO’
    [ 15]    ’NeuronId’
    [ 16]    ’TracesetIndex’
...
```

One can choose which parameters need to be distinct for each row by specifying as the second argument in the call to the `meanDuplicateRows` method, whereas the third argument specifies the measures to be averaged:

```
>> phys_mean_db = meanDuplicateRows(phys_db, [4 6:15], [17:161])
params_tests_db, averaged tracesets from .../cip_traces_all.txt
ans =
    num_params: 13
         props: [0x0 struct]
      tests_db: [1x1 tests_db]
Optional properties of params_tests_db:
ans =
0x0 struct array with no fields.
```

```
tests_db, averaged tracesets from .../cip_traces_all.txt
690 rows in database with 158 columns, and 1 pages.
Column names:
    [  1]     'pAcip'
    [  2]     'Cadmium'
    [  3]     'PicroTx'
    [  4]     'Apamin'
    [  5]     'Glycine'
    [  6]     'KynAcid'
    [  7]     'TTX'
    [  8]     'XE991'
    [  9]     'drug_4AP'
    [ 10]     'EBIO'
    [ 11]     'NeuronId'
    [ 12]     'NumDuplicates'
    [ 13]     'RowIndex'
    [ 14]     'IniSpontISICV'
...
```

This command ignores the pulse time information and the bias current, but includes all the pharmacological parameters, in distinguising the unique traces. It also eliminates some measures, such as the `ItemIndex`, from averaging.

5.5 Making a database by merging multiple rows from another database

A simple example for making a new database out of multiple rows in an existing database is combining multiple traces from the same neuron with different current pulse injection (CIP) levels. The initial database contains a row for each CIP level with redundant information, such as spontaneous period measurements.

5.5.1 Making a one-row-per-neuron DB from multiple CIP-level rows

Measures of same cell obtained with multiple CIP-levels can be merged to make a single row. Note that, different pharmacological conditions applied to one cell must be kept in a different rows. The following command selects measures from each of the $\{-100, 0, 40, 100, 200\}$ CIP levels to be included in the merged database:

```
>> phys_joined_db =
     mergeMultipleCIPsInOne(phys_mean_db(:, [1:13 16:180]),
                           {'_H100pA', 13 + [5:14 19:24 (119 + spike_tests) 165],
                            '_0pA', 13 + [1:4 (27 + spike_tests)],
                            '_D40pA', 13 + [5:11 19:24 (73 + spike_tests) 165],
                            '_D100pA', 13 + [5:11 14:16 19:24 (73 + spike_tests)
                                             (119 + spike_tests) 165],
                            '_D200pA', 13 + [5:11 19:24 (73 + spike_tests) 165]},
                           'RowIndex_D200pA')
```

This command operates on the previously averaged database, `phys_mean_db`, where each CIP level only occurs once for each distinct pharmacological setting for each neuron. Note that, we filter-out columns 14 and 15 from the averaged DB while supplying the first argument to `mergeMultipleCIPsInOne`, which are artifacts of the averaging process and need not be included in the merged database. The second argument is a cell array of pairs of a suffix string and a corresponding list of measures for each of the CIP levels in `phys_mean_db`, in increasing order. The merged `phys_joined_db` looks like this:

```
params_tests_db, averaged tracesets from .../cip_traces_all_axoclamp.txt mult CIP
ans =
    num_params: 12
      tests_db: [1x1 tests_db]
tests_db, averaged tracesets from .../cip_traces_all_axoclamp.txt mult CIP
179 rows in database with 258 columns, and 1 pages.
Column names:
    [  1]    ’Cadmium’
    [  2]    ’PicroTx’
    [  3]    ’Apamin’
    [  4]    ’Glycine’
    [  5]    ’KynAcid’
    [  6]    ’TTX’
    [  7]    ’XE991’
    [  8]    ’drug_4AP’
    [  9]    ’EBIO’
    [ 10]    ’Gabazine’
    [ 11]    ’NeuronId’
    [ 12]    ’TracesetIndex’
    [ 13]    ’PulseISICV_H100pA’
    [ 14]    ’PulseIni100msISICV_H100pA’
    [ 15]    ’PulseIni100msRest1SpikeRate_H100pA’
    [ 16]    ’PulseIni100msRest2SpikeRate_H100pA’
    [ 17]    ’PulseIni100msSpikeRate_H100pA’
    [ 18]    ’PulseIni100msSpikeRateISI_H100pA’
    ...
    [255]    ’PulseSpikeRiseTimeMean_D200pA’
    [256]    ’PulseSpikeRiseTimeMode_D200pA’
    [257]    ’PulseSpikeRiseTimeSTD_D200pA’
    [258]    ’PulseSpontAmpRatio_D200pA’
```

Note how each measure suffix indicate the CIP-level it belongs.

5.5.2 Making a one-row-per-neuron DB from dual CIP-level rows

The following statement uses the `params_tests_db/getDualCIPdb` method to merge rows of depolarizing and hyperpolarizing CIP-levels:

```
>> sdball = getDualCIPdb(dball, depol_tests, hyper_tests,
                         ”, ’Hyp100pA’)
```

Here, the cell array variables `depol_tests` and `hyper_tests` hold the names of measures to be selected from depolarizing CIP and hyperpolarizing CIP, respectively. The last two arguments define the suffixes to be applied to distinguish the measures from each CIP. The original DB is

```
>> dball
dball
params_tests_db, sim dataset gpsc0501
ans =
    num_params: 10
         props: [0x0 struct]
      tests_db: [1x1 tests_db]
Optional properties of params_tests_db:
ans =
0x0 struct array with no fields.
tests_db, sim dataset gpsc0501
39366 rows in database with 62 columns, and 1 pages.
Column names:
    [ 1]    ’NaF’
    [ 2]    ’NaP’
    [ 3]    ’Kv3’
    [ 4]    ’Kv2’
    [ 5]    ’Kv4f’
    [ 6]    ’KCNQ’
    [ 7]    ’SK’
    [ 8]    ’CaHVA’
    [ 9]    ’HCN’
    [10]    ’pAcip’
    [11]    ’IniSpontISICV’
    [12]    ’IniSpontPotAvg’
    [13]    ’IniSpontSpikeRate’
    [14]    ’PulseISICV’
    [15]    ’PulseIni100msISICV’
    [16]    ’PulseIni100msRest1SpikeRate’
    [17]    ’PulseIni100msRest2SpikeRate’
    [18]    ’PulseIni100msSpikeRate’
    [19]    ’PulseIni100msSpikeRateISI’
    [20]    ’PulsePotAvg’
    [21]    ’PulsePotMin’
    [22]    ’PulsePotSag’
    [23]    ’PulseSFA’
    [24]    ’PulseSpikeAmpDecayDelta’
    [25]    ’PulseSpikeAmpDecayTau’
    [26]    ’PulseSpikeRate’
    [27]    ’PulseSpikeRateISI’
    [28]    ’RecIniSpontPotRatio’
    [29]    ’RecIniSpontRateRatio’
    [30]    ’RecSpont1SpikeRate’
    [31]    ’RecSpont2SpikeRate’
    [32]    ’RecSpontISICV’
    [33]    ’RecSpontPotAvg’
    [34]    ’RecSpontSpikeRate’
    [35]    'SpontAmplitude'
    [36]    'SpontBaseWidth'
    [37]    'SpontDAHPMag'
    [38]    'SpontFallTime'
    [39]    'SpontHalfVm'
    [40]    'SpontHalfWidth'
    [41]    'SpontInitTime'
    [42]    'SpontInitVm'
    [43]    'SpontMaxAHP'
    [44]    'SpontMinTime'
    [45]    'SpontMinVm'
    [46]    'SpontPeakVm'
    [47]    'SpontRiseTime'
    [48]    'PulseAmplitude'
    [49]    'PulseBaseWidth'
    [50]    'PulseDAHPMag'
    [51]    'PulseFallTime'
    [52]    'PulseHalfVm'
    [53]    'PulseHalfWidth'
    [54]    'PulseInitTime'
    [55]    'PulseInitVm'
    [56]    'PulseMaxAHP'
    [57]    'PulseMinTime'
    [58]    'PulseMinVm'
    [59]    'PulsePeakVm'
    [60]    'PulseRiseTime'
    [61]    'PulseSpontAmpRatio'
    [62]    'ItemIndex'
Optional properties of tests_db:
ans =
0x0 struct array with no fields.
```

After merging, it becomes

```
>> sdball
sdball
params_tests_db, sim dataset gpsc0501 dual cip
ans =
    num_params: 9
         props: [0x0 struct]
      tests_db: [1x1 tests_db]
Optional properties of params_tests_db:
ans =
0x0 struct array with no fields.
tests_db, sim dataset gpsc0501 dual cip
19683 rows in database with 50 columns, and 1 pages.
Column names:
    [ 1]    'NaF'
    [ 2]    'NaP'
    [ 3]    'Kv3'
    [ 4]    'Kv2'
    [ 5]    ’Kv4f’
    [ 6]    ’KCNQ’
    [ 7]    ’SK’
    [ 8]    ’CaHVA’
    [ 9]    ’HCN’
    [10]    ’RecIniSpontPotRatioHyp100pA’
    [11]    ’RecIniSpontRateRatioHyp100pA’
    [12]    ’RecSpont1SpikeRateHyp100pA’
    [13]    ’RecSpont2SpikeRateHyp100pA’
    [14]    ’RecSpontISICVHyp100pA’
    [15]    ’RecSpontPotAvgHyp100pA’
    [16]    ’ItemIndexHyp100pA’
    [17]    ’IniSpontSpikeRate’
    [18]    ’PulseIni100msSpikeRate’
    [19]    ’PulseIni100msSpikeRateISI’
    [20]    ’PulseIni100msISICV’
    [21]    ’PulseIni100msRest1SpikeRate’
    [22]    ’PulseIni100msRest2SpikeRate’
    [23]    ’RecSpont1SpikeRate’
    [24]    ’RecSpont2SpikeRate’
    [25]    ’RecIniSpontRateRatio’
    [26]    ’IniSpontISICV’
    [27]    ’PulseISICV’
    [28]    ’RecSpontISICV’
    [29]    ’PulseSFA’
    [30]    ’PulseSpikeAmpDecayTau’
    [31]    ’PulseSpikeAmpDecayDelta’
    [32]    ’IniSpontPotAvg’
    [33]    ’PulsePotAvg’
    [34]    ’RecSpontPotAvg’
    [35]    ’RecIniSpontPotRatio’
    [36]    ’SpontInitVm’
    [37]    ’SpontAmplitude’
    [38]    ’SpontMaxAHP’
    [39]    ’SpontDAHPMag’
    [40]    ’SpontRiseTime’
    [41]    ’SpontFallTime’
    [42]    ’SpontHalfWidth’
    [43]    ’PulseInitVm’
    [44]    ’PulseAmplitude’
    [45]    ’PulseMaxAHP’
    [46]    ’PulseDAHPMag’
    [47]    ’PulseRiseTime’
    [48]    ’PulseFallTime’
    [49]    ’PulseHalfWidth’
    [50]    ’ItemIndex’
Optional properties of tests_db:
ans =
0x0 struct array with no fields.
```

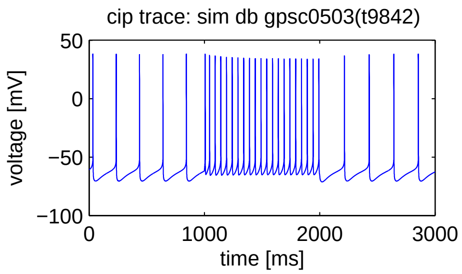


Figure 8: Example trace plot.

6 Visualization

By default, MATLAB prints figures in portrait orientation with a 8x6 aspect ratio, ignoring their size on the screen. The command;

```
>> orient tall
```

changes that behavior to print a full page in portrait orientation. The command;

```
>> orient landscape
```

does the same but rotates the figure to landscape orientation. If you want the printed figure to reflect its current screen size, issue the command;

```
>> set(figurenum, 'PaperPositionMode', 'auto')
```

Figures generated by the plotting system of the PANDORA Toolbox has a special resizing capability. Everytime the figure is resized, it will be drawn from scratch after calculating proper spacing between subplots according to font size. However, in some conditions this may cause other problems, such as crashing MATLAB in figure editing mode or causing loss of manual changes to the figure. To disable the auto-resize function, issue the following command after creating the figure;

```
>> set(figurenum, 'ResizeFcn', ”)
```

Finally, to print the figure, consult your Matlab manual or issue a command such as;

```
>> print -depsc2 figurename.eps
```

See below for specific types of figures you can create.

6.1 Visualizing traces

The `trace,` and its subclasses such as `cip_trace`, has the plot command overloaded to draw the raw trace in Figure 8:

```
>> a_ct = ctFromRows(mbundle, 9842, 100);
>> plot(a_ct)
```

Note that the plot in the figure has been created with a more precise control:

```
>> plotFigure(plotData(a_ct, ”, struct('PaperPosition', [0 0 3 2])))
```

6.2 Displaying database contents

The displayRows method of tests_db can be used to display rows of a database:

```
>> displayRows(sdball, 1:3)
ans =
    'NaF'                    [      250]   [      250]   [      1000]
    'NaP'                    [   2.5000]   [   2.5000]   [    2.5000]
    'Kv3'                    [       15]   [       15]   [        60]
    'Kv2'                    [        3]   [        3]   [         9]
    'Kv4f'                   [        5]   [        5]   [        25]
    'KCNQ'                   [   0.1000]   [   0.1000]   [    0.0100]
    'SK'                     [   8.5000]   [   8.5000]   [        17]
    'CaHVA'                  [       10]   [       10]   [        10]
    'HCN'                    [       30]   [        3]   [        30]
    'pAcip'                  [     -100]   [     -100]   [       100]
    'IniSpontISICV'          [   0.0027]   [   0.0027]   [9.1376e-04]
...
```

Note that, in the output, database rows appear as columns, and database columns appear as rows. See in above Section 5.3 for more example outputs from displayRows.

displayRows returns a cell array of column names justapoxed to a matrix of values. This cell array is intended for display on the screen and for generating reports. The displayRowsTeX method uses output from displayRows to generate a LaTeX table that can be printed or converted to PDF:

```
>> tex_string =
     displayRowsTeX(a_db(:, rows),
                    'Selected rows indicating fast spiking neurons',
                    struct('height', '!'));
>> string2File(tex_string, 'fast_spiking.tex');
```

With this the LaTeX code to generate a table with the given caption is saved in a text file called fast_spiking.tex. This file can then be included from a regular LaTeX document to generate PDF output. See a LaTeX manual on how to do that.

An alternative to displayRows is using the tests_db/rows2Struct method:

```
>> s = rows2Struct(dball_full, 54023)
s =
                            NaF: 500
                            NaP: 2
                            Kv2: 1
                            Kv3: 10
                           Kv4f: 40
                           KCNQ: 2
                             SK: 4
                          CaHVA: 0.0300
                            HCN: 1
                          trial: 7739
                          pAcip: 100
                  IniSpontISICV: NaN
                 IniSpontPotAvg: NaN
```

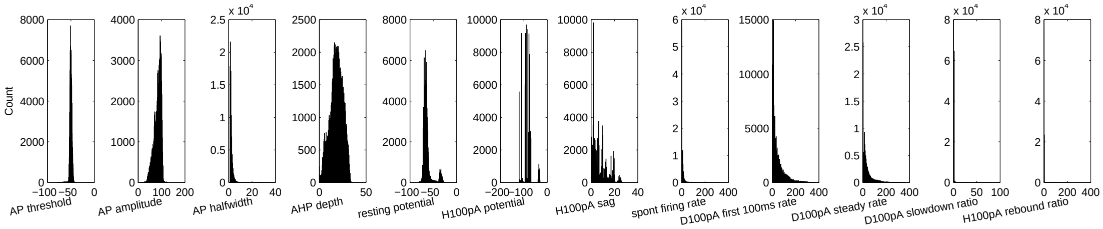


Figure 9: Example measure distribution plot.

```
          IniSpontSpikeRate: 0
       IniSpontSpikeRateISI: 0
                 PulseISICV: 0.0265
         PulseIni100msISICV: 0.0584
PulseIni100msRest1SpikeRate: 40.0089
PulseIni100msRest2SpikeRate: 40.0178
     PulseIni100msSpikeRate: 50
  PulseIni100msSpikeRateISI: 43.5256
                PulsePotAvg: -57.3737
                PulsePotMin: NaN
                PulsePotSag: NaN
                   PulseSFA: 1.1698
...
```

This method returns the database contents as a structure array. It is more natural for programming interfaces to use the database contents in a structure array than a cell array. The database columns become field names in the the structure. If multiple rows are requested, the displayed output would not contain the values. The desired row can be reached via indexing (e.g., `s(1)`). For instance, analysis in `cip_trace/getProfileAllSpikes` method is done using this method for getting statistics from the `spikes_db` databases.

6.3 Plotting all measure histograms

For plotting all measure histograms in a DB, the following method of `tests_db` creates the horizontal stack plot in Figure 9:

```
>> mp = plotTestsHistsMatrix(renamed_model_db(:, 2:end), ”,
                             struct(’orient’, ’x’, ’quiet’, 1, ’border’, [0 0.05 0 0]))
>> plotFigure(mp);
```

6.4 Plotting all parameter histograms

For plotting all parameter histograms in a DB, the following method of `params_tests_db` creates the horizontal stack plot in Figure 10:

```
>> plotFigure(plotParamsHists(sdball));
```

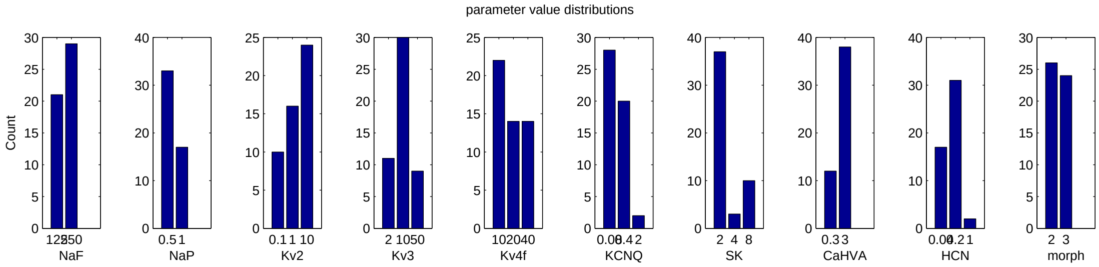


Figure 10: Example parameter distribution plot.

6.5 Plotting database statistics

The `stats_db` object allows keeping statistical information obtained from a database. Statistics are calculated using one of the `tests_db` converter methods, such as `statsAll`, `statsMeanStd`, etc.:

```
>> my_stats = statsMeanStd(my_db(:, {'IniSpontSpikeRate', 'PulseSpikeRate'}));
```

Then, the statistics can be plotted with diamonds indicating the mean and symmetric errorbars indicating upper and lower extensions (SE or Std):

```
>> plot(my_stats);
```

which is equivalent to:

```
>> plotFigure(plot_abstract(my_stats));
```

An alternative plotting form is using filled bars with extending errorbars:

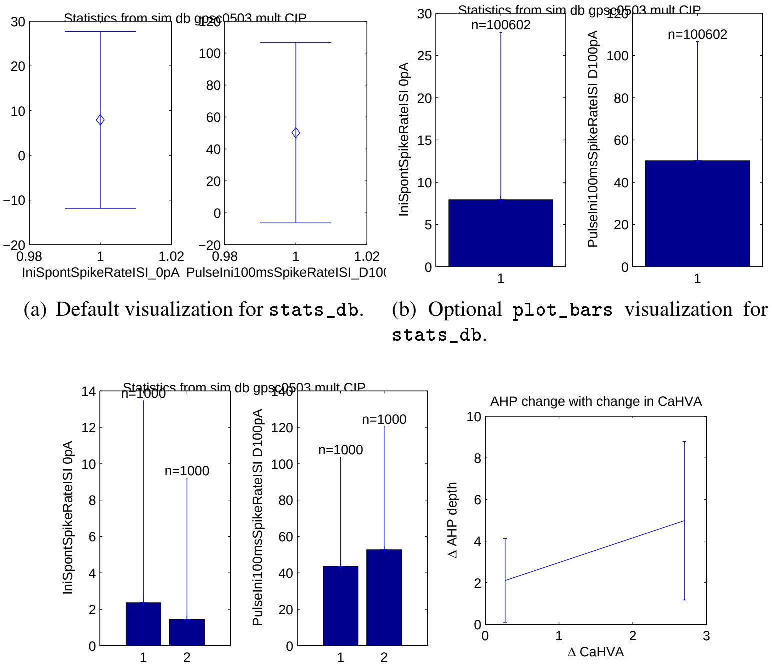
```
>> plotFigure(plot_bars(my_stats));
```

which are both seen in Figure 11 (a) and (b). The `plot_bars` method is better suited for comparing statistics from multiple `stats_dbs` coming from different sources. In this case, we take the first 1000 rows from the DB as a subset, and compare it with the statistics from the second 1000 rows:

```
>> compared_two_subset_stats =
     compareStats(statsAll(mbundle.joined_db(1:1000,
                                          {'IniSpontSpikeRateISI_0pA',
                                           'PulseIni100msSpikeRateISI_D100pA'})),
                 statsAll(mbundle.joined_db(1001:2000,
                                          {'IniSpontSpikeRateISI_0pA',
                                           'PulseIni100msSpikeRateISI_D100pA'}))));
```

The combined statistics object can then be fed into `plot_bars` as seen in Figure 11 (c):

```
>> plotFigure(plot_bars(compared_two_subset_stats));
```



(a) Default visualization for `stats_db`.

(b) Optional `plot_bars` visualization for `stats_db`.

(c) Comparing multiple `stats_dbs` with `plot_bars`.

(d) Plotting parameter-measure variation with `plotVar`.

Figure 11: Plotting statistics for two selected measures.

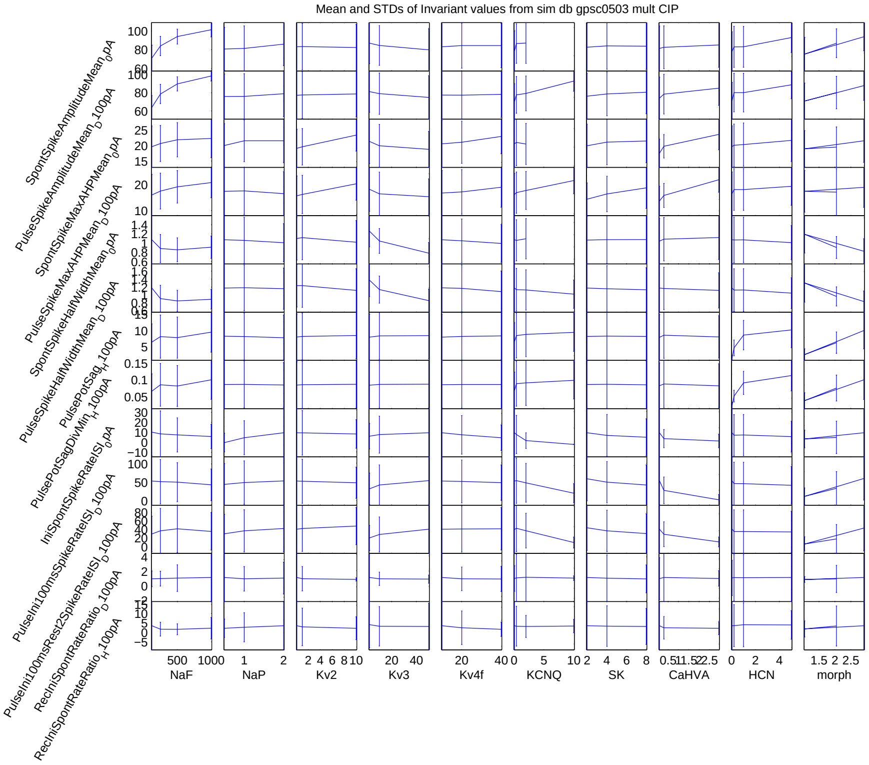


Figure 12: Example parameter-measure variation statistics plot.

6.6 Plotting parameter-measure variations

To plot the variation of a measure with a parameter the `plotVar` method of `stats_db` can be used to achieve Figure 11 (d):

```
>> plotFigure(plotVar(a_stats_db, 'CaHVA', 'AHP_depth', 'AHP change with change in CaHVA',
                    struct('quiet', 1, 'PaperPosition', [0 0 3 3])))
```

To plot all parameter-measure variations, the `plotVarMatrix` method of `stats_db` can be used (see Figure 12). `plotVarMatrix` requires the a `p_stats` array of `stats_db` objects that hold the mean and standard error information (or possibly other statistics) for each of the possible parameter-measure combinations. The `p_stats` array can be created using the `paramsTestsHistsStats` method, which in turn requires the `p_t3ds` array of 3-dimensional databases each of which contain effects of a parameter invariant of other parameters. The `p_t3ds` array can be created using the `params_tests_db/invarParams` method. The sequence of commands[4] is then becomes:

```
>> p_t3ds = invarParams(noNaNRows(sdball))
>> [pt_hists, p_stats] = paramsTestsHistsStats(p_t3ds)
>> ap = plotVarMatrix(p_stats)
>> plotFigure(ap)
```

This will create a matrix plot with as many columns as parameters and as many rows as measures in the `sdball` object. It may be difficult to read if `sdball` contains large number of measures. One can divide the measures into two plots with the following sequence of commands

```
>> sdb1 = sdball(:, [1:9, 10:35])
>> sdb2 = sdball(:, [1:9, 36:49])
```

by choosing all parameters in both DBs, but only some measures for each. Then, the plots can be created for each DB by issuing

```
>> p1_t3ds = invarParams(sdb1)
>> p2_t3ds = invarParams(sdb2)
>> [pt1_hists, p1_stats] = paramsTestsHistsStats(p1_t3ds)
>> [pt2_hists, p2_stats] = paramsTestsHistsStats(p2_t3ds)
>> ap1 = plotVarMatrix(p1_stats)
>> plotFigure(ap1)
>> ap2 = plotVarMatrix(p2_stats)
>> plotFigure(ap2)
```

6.7 Insets

See Figure 13 which was created with the following set of commands.

```
>> im_p = plot_abstract({50 * rand(5)}, {}, ”, {}, 'image');
>> plotFigure(plot_absolute([im_p, im_p], [0 0 1 1; 0.5 0.5 0.3 0.3]))
```

[4] The `noNaNRows` function is required to filter out any rows containing `NaN` values in measurements before running statistic functions. Otherwise, the statistics functions (such as `mean` and `std`) will eliminate `NaN` values within each database column automatically, scrambling the row order and loosing the association with parameters.

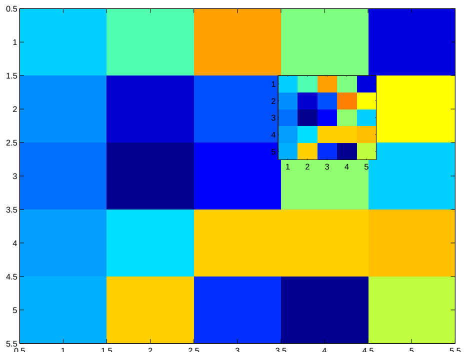


Figure 13: Creating insets in figures.

6.8 Generating a report comparing two databases

Most commonly a database of physiology neurons need to be compared to a database of a large body of simulation neurons and find best matches. One may need to see the match quality for a number of best matching candidates for each of the physiology neurons. Assuming multiple plots and tables are required to do a fair job of comparing a neuron to a thousands of simulation neurons, it becomes a difficult job to do this manually. An automatic report generation system has been built into the system for this purpose.

Currently a LaTeX document is created that needs to be included in either a proper LaTeX document, or included in a LyX document.[5] The including document should provide the context in which the included part becomes meaningful. The report contains a set of tables and figures with proper captions. The table of contents, list of figures and list of tables facilities of LaTeX becomes useful to make the automatically generated document easily readable.

The report can be created with the command

```
>> tex_string = rankVsAllDB(sdb, phys_sdb,
                            fileset, phys_fileset);
```

where the simulation database `sdb` is searched for best matches to each row of physiology database `phys_sdb`. The dataset for each is provided for the report to contain raw data associated with best matches. The obtained report contained in `tex_string` can be saved as a ASCII LaTeX file with

```
>> string2File(tex_string, 'myreport.tex');
```

[5] This document is prepared using the LyX document preparation system [Ettrich et al., 2003] which uses the LaTeX 2_ε typesetting language [Lamport, 1994]. LyX is copyrighted by Matthias Ettrich and covered by the terms of the GNU General Public License (GPL), and LaTeX 2_ε is copyrighted by D. E. Knuth and the Free Software Foundation, Inc. and is covered by both the TeX copyright and the GNU GPL.

References

Matthias Ettrich et al. LyX. A document preparation system that allows the author to concentrate on content, rather than typesetting., 2003. URL http://www.lyx.org.

Leslie Lamport. *LaTeX: A document preparation system*. Addison-Wesley, Reading, Massachusetts, second edition, 1994. ISBN 0-201-52983-1.

Appendices

A Function Reference

See Section 1.6 on how to get online help about the software within MATLAB.

A.1 Class `chans_db`

A.1.1 Constructor `chans_db/chans_db`

Summary: A database of channel activation and kinetics.

Usage:

```
a_chans_db = chans_db(data, col_names, channel_info, id, props)
```

Description: This is a subclass of tests_db. Channel tables can be imported from Genesis using the utils/chanTables2DB script.

Parameters:

`data`**:** Database contents.

`col_names`**:** The channel variable names.

`channel_info`**:** Structure that holds scalar data elements such as Gbar.

`id`**:** An identifying string.

`props`**:** A structure with any optional properties.

Returns a structure object with the following fields:

tests_db, channel_info, props.

See also: `tests_db` (p. 227), `chanTables2DB` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/06/26

A.1.2 Method chans_db/get

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.1.3 Method chans_db/set

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.1.4 Method chans_db/plotInf

Summary: Plot the product of minf variables and the gmax of the given channel.

Usage:

```
a_plot = plotInf(a_chans_db, chan_name, gate_subnames, title_str, props)
```

Parameters:

`a_chans_db`**:** A chans_db describing channel variables.

`chan_name`**:** Name of channel that make up the stem of variable names.

`gate_subnames`**:** Gate names of the channel.

`title_str`**:** (Optional) A string to be concatanated to the title.

`props`**:** A structure with any optional properties.
(rest passed to plot_abstract.)

Returns:

a_plot: A plot_abstract object that can be visualized.

See also: `trace` (p. 278), `trace/plot` (p. 280), `plot_abstract` (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/07/01

A.1.5 Method chans_db/display

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.1.6 Method chans_db/plotAllInf

Summary: Plots the steady-state (infinity) response of all channels.

Usage:

```
a_plot = plotAllInf(a_chans_db, title_str, props)
```

Parameters:

a_chans_db: a chans_db

title_str: Plot title.

props: A structure with any optional properties.
(rest passed to matrixPlots.)

Returns:

a_plot: A plot_abstract object that can be visualized.

See also: trace (p. 278), trace/plot (p. 280), plot_abstract (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/03/05

A.1.7 Method chans_db/plotAllVars

Summary: Plot all channel variables by grouping activation and time constant curves per channel.

Usage:

```
a_plot = plotAllVars(a_chans_db, title_str, props)
```

Parameters:

a_chans_db: A chans_db describing channel variables.

id: String that identify the source of the tables structure.

props: A structure with any optional properties.
(rest passed to plot_abstract.)

Returns:

a_plot: A plot_abstract object that can be visualized.

See also: trace (p. 278), trace/plot (p. 280), plot_abstract (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/03/05

A.1.8 Method chans_db/plotGateVars

Summary: Plot given channel gate variables of the same channel superposed.

Usage:

```
a_plot = plotGateVars(a_chans_db, chan_name, gate_subnames, title_str, props)
```

Parameters:

a_chans_db: A chans_db describing channel variables.

chan_name: Name of channel that make up the stem of variable names.

gate_subnames: Gate names of the channel.

title_str: (Optional) A string to be concatanated to the title.

props: A structure with any optional properties.

usePowers: Use the gate powers, Luke.
(rest passed to plot_abstract.)

Returns:

a_plot: A plot_abstract object that can be visualized.

See also: trace (p. 278), trace/plot (p. 280), plot_abstract (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/07/01

A.2 Class cip_trace

A.2.1 Constructor cip_trace/cip_trace

Summary: A trace with a current injection pulse (CIP).

Usage:

```
obj = cip_trace(datasrc, dt, dy, pulse_time_start, pulse_time_width, id, props)
```

Parameters:

datasrc: A vector of data points containing the spike shape.

dt: Time resolution [s].

dy: y-axis resolution [ISI (V, A, etc.)]

pulse_time_start, pulse_time_width: Start and width of the pulse [dt]

id: Identification string.

props: A structure with any optional properties, such as:

trace_time_start: Samples in the beginning to discard [dt]
(see trace for more)

Returns a structure object with the following fields:

trace, pulse_time_start, pulse_time_width, props.

See also: `trace` (p. 278), `spikes` (p. 202), `spike_shape` (p. 189), `period` (p. 140)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/07/30

A.2.2 Method `cip_trace/getPulseSpike`

Summary: Convert a spike in the CIP period to a spike_shape object.

Usage:

```
obj = getPulseSpike(trace, spikes, spike_num, props)
```

Description: Creates a spike_shape object from desired spike. Calls trace/getSpike method.

Parameters:

`trace`**:** A trace object.

`spikes`**:** (Optional) A spikes object obtained from trace,
calculated automatically if given as [].

`spike_num`**:** The index of spike to extract.

`props`**:** A structure with any optional properties passed to getSpike.

Example:

```
Get 2nd pulse spike and plot it:
» plotFigure(plotResults(getPulseSpike(t, [], 2)))
```

See also: `spike_shape` (p. 189), `trace/getSpike` (p. 287)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/04/19

A.2.3 Method `cip_trace/plotData`

Summary: Plots a trace by calling trace/plotData but also adds optionaldecorations.

Usage:

```
a_plot = plotData(t, title_str, props)
```

Description: If t is a vector of traces, returns a vector of plot objects.

Parameters:

`t`**:** A trace object.

`title_str`**:** (Optional) String to append to plot title.

`props`**:** A structure with any optional properties.

`stimBar`**:** If true, put a bar indicating the CIP duration.
(rest passed to trace/plotData)

Returns:

a_plot: A plot_abstract object that can be visualized.

See also: `trace` (p. 278), `trace/plot` (p. 280), `plot_abstract` (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/17

A.2.4 Method `cip_trace/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.2.5 Method `cip_trace/periodRecSpont2`

Summary: Returns the second half of the recovery spontaneous activity period of cip_trace, t.

Usage:

```
the_period = periodRecSpont2(t)
```

Parameters:

`t`**:** A trace object.

Returns:

the_period: A period object.

See also: `period` (p. 140), `cip_trace` (p. 56), `trace` (p. 278)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/25

A.2.6 Method cip_trace/periodRecSpontRestPeriod

Usage:

```
the_period = periodRecSpont(t)
```

Parameters:

t: A trace object.

iniPeriod: the time following pulse offset that is ignored. The rest of the time is kept

Returns:

the_period: A period object.

See also: period (p. 140), cip_trace (p. 56), trace (p. 278)

Author: Cengiz Gunay <cgunay@emory.edu>,Tom Sangrey 2006/01/26

A.2.7 Method cip_trace/periodPulse

Summary: Returns the CIP period of cip_trace, t.

Usage:

```
the_period = periodPulse(t)
```

Parameters:

t: A trace object.

Returns:

the_period: A period object.

See also: period (p. 140), cip_trace (p. 56), trace (p. 278)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/25

A.2.8 Method cip_trace/getResults

Summary: Calculate test results given a_spikes object.

Usage:

```
results = getResults(a_cip_trace, a_spikes)
```

Parameters:

a_cip_trace: A cip_trace object.

a_spikes: A spikes object.

Returns:

results: A structure associating test names with result values.

See also: cip_trace (p. 56), spikes (p. 202), spike_shape (p. 189)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.2.9 Method cip_trace/set

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.2.10 Method cip_trace/periodPulseIni100msRest1

Usage:

```
the_period = periodPulseIni50msRest1(t)
```

Parameters:

t: A trace object.

Returns:

the_period: A period object.

See also: period (p. 140), cip_trace (p. 56), trace (p. 278)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/25

A.2.11 Method cip_trace/periodIniSpont

Summary: Returns the initial spontaneous activity period of cip_trace, t.

Usage:

```
the_period = periodIniSpont(t)
```

Parameters:

t: A trace object.

Returns:

the_period: A period object.

See also: period (p. 140), cip_trace (p. 56), trace (p. 278)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/25

A.2.12 Method `cip_trace/periodPulseIni100ms`

Summary: Returns the first 100ms of the CIP period of cip_trace, t.

Usage:

```
the_period = periodPulseIni100ms(t)
```

Parameters:

`t`**:** A trace object.

Returns:

the_period: A period object.

See also: `period` (p. 140), `cip_trace` (p. 56), `trace` (p. 278)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/25

A.2.13 Method `cip_trace/getCIPResults`

Summary: Calculate test results about CIP protocol.

Usage:

```
results = getCIPResults(a_cip_trace, a_spikes)
```

Parameters:

`a_cip_trace`**:** A cip_trace object.

`a_spikes`**:** A spikes object.

Returns:

results: A structure associating test names with result values.

See also: `cip_trace` (p. 56), `spikes` (p. 202), `spike_shape` (p. 189)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/30

A.2.14 Method `cip_trace/periodPulseHalf1`

Summary: Returns the first half of the CIP period of cip_trace, t.

Usage:

```
the_period = periodPulseHalf1(t)
```

Parameters:

`t`**:** A trace object.

Returns:

the_period: A period object.

See also: period (p. 140), cip_trace (p. 56), trace (p. 278)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/25

A.2.15 Method cip_trace/periodRecSpont1

Summary: Returns the first half of the recovery spontaneous activity period of cip_trace, t.

Usage:

```
the_period = periodRecSpont1(t)
```

Parameters:

t: A trace object.

Returns:

the_period: A period object.

See also: period (p. 140), cip_trace (p. 56), trace (p. 278)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/25

A.2.16 Method cip_trace/subsref

Summary: Defines generic indexing for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.2.17 Method cip_trace/getBurstResults

Summary: Calculate test results related to Burst behavior.

Usage:

```
results = getRateResults(a_cip_trace, a_spikes)
```

Parameters:

a_cip_trace: A cip_trace object.

a_spikes: A spikes object.

Returns:

results: A structure associating test names with result values.

See also: `cip_trace` (p. 56), `spikes` (p. 202), `spike_shape` (p. 189)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/30, Tom Sangrey

A.2.18 Method `cip_trace/periodPulseIni50msRest1`

Summary: Returns the first half of the rest after the first 50ms of the CIP period of cip_trace, t.

Usage:

```
the_period = periodPulseIni50msRest1(t)
```

Parameters:

`t`**:** A trace object.

Returns:

the_period: A period object.

See also: `period` (p. 140), `cip_trace` (p. 56), `trace` (p. 278)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/25

A.2.19 Method `cip_trace/measureNames`

Summary: Returns taxonomy of measurements collected by cip_trace.

Usage:

```
measures = measureNames(a_cip_trace)
```

Description: This is a static method, in the sense that it does not need the object passed as argument. Therefore it can be called directly by using the default constructor; e.g., measureNames(cip_trace). The measure names are required for merging columns of a database generated by profiling these objects.

Parameters:

`a_cip_trace`**:** A cip_trace object. It can be created by the the default constructor 'cip_trace'.

Returns:

measures: A structure with cell arrays of types of measures, and measure names inside.

See also: getResults (p. **??**), getProfileAllSpikes (p. **??**), mergeMultipleCIPsInOne (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/07/30

A.2.20 Method `cip_trace/display`

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.2.21 Method `cip_trace/calcRecSpontPotAvg`

Summary: Calculates the average potential value of the recovery period of the cip_trace, t.

Usage:

```
avg_val = calcRecSpontPotAvg(t)
```

Parameters:

`t`**:** A cip_trace object.

Returns:

avg_val: The avg value [dy].

See also: period (p. 140), trace (p. 278), trace/calcAvg (p. 282)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/25

A.2.22 Method `cip_trace/getRateResults`

Summary: Calculate test results related to spike rate.

Usage:

```
results = getRateResults(a_cip_trace, a_spikes)
```

Parameters:

`a_cip_trace`**:** A cip_trace object.

`a_spikes`**:** A spikes object.

Returns:

results: A structure associating test names with result values.

See also: cip_trace (p. 56), spikes (p. 202), spike_shape (p. 189)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/30

A.2.23 Method `cip_trace/calcPulsePotAvg`

Summary: Calculates the average potential value of the CIP period of the cip_trace, t.

Usage:

```
avg_val = calcPulsePotAvg(t)
```

Parameters:

`t`**:** A cip_trace object.

Returns:

avg_val: The avg value [dy].

See also: `period` (p. 140), `trace` (p. 278), `trace/calcAvg` (p. 282)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/25

A.2.24 Method `cip_trace/periodRecSpontIniPeriod`

Usage:

```
the_period = periodRecSpont(t)
```

Parameters:

`t`**:** A trace object.

`iniPeriod`**:** the time following pulse offset that is kept, the rest of the time is ignored.

Returns:

the_period: A period object.

See also: `period` (p. 140), `cip_trace` (p. 56), `trace` (p. 278)

Author: Cengiz Gunay <cgunay@emory.edu>,Tom Sangrey 2006/01/26

A.2.25 Method `cip_trace/periodPulseIni100msRest2`

Usage:

```
the_period = periodPulseIni50msRest2(t)
```

Parameters:

`t`**:** A trace object.

Returns:

the_period: A period object.

See also: `period` (p. 140), `cip_trace` (p. 56), `trace` (p. 278)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/25

A.2.26 Method `cip_trace/getRecSpontSpike`

Summary: Convert a spike in the CIP period to a spike_shape object.

Usage:

```
obj = getRecSpontSpike(trace, spikes, spike_num, props)
```

Description: Creates a spike_shape object from desired spike.

Parameters:

`trace`: A trace object.

`spikes`: A spikes object on trace.

`spike_num`: The index of spike to extract.

See also: `spike_shape` (p. 189)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/05/08

A.2.27 Method `cip_trace/spikes`

Summary: Convert cip_trace to spikes object for spike timing calculations.

Usage:

```
obj = spikes(trace, plotit)
```

Description: Creates a spikes object by finding the spikes in the three separate periods, initial spontaneous activity period, CIP period, and final recovery period.

Parameters:

`trace`: A trace object.

`plotit`: If non-zero, a plot is generated for showing spikes found (optional).

See also: `spikes` (p. 202), `period` (p. 140)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/25

A.2.28 Method cip_trace/periodPulseIni50msRest2

Summary: Returns the second half of the rest after the first 50ms of the CIP period of cip_trace, t.

Usage:

```
the_period = periodPulseIni50msRest2(t)
```

Parameters:

t: A trace object.

Returns:

the_period: A period object.

See also: period (p. 140), cip_trace (p. 56), trace (p. 278)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/25

A.2.29 Method cip_trace/periodRecSpont

Summary: Returns the recovery spontaneous activity period of cip_trace, t.

Usage:

```
the_period = periodRecSpont(t)
```

Parameters:

t: A trace object.

Returns:

the_period: A period object.

See also: period (p. 140), cip_trace (p. 56), trace (p. 278)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/25

A.2.30 Method cip_trace/calcPulsePotSag

Summary: Calculates the minimal sag and sag amount of the CIP period of the cip_trace, t.

Usage:

```
[min_val, min_idx, sag_val] = calcPulsePotSag(t)
```

Description: The minimal sag is the minimal potential value of the first half of the CIP period. The sag amount is calculated by comparing this to the steady-state value at the end of the CIP period.

Parameters:

`t`: A cip_trace object.

Returns:

min_val: The min value [dy]. min_idx: The index of the min value [dt]. sag_val: The sag amount [dy].

See also: `period` (p. 140), `trace` (p. 278), `trace/calcMin` (p. 284)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/25

A.2.31 Method `cip_trace/periodPulseIni50ms`

Summary: Returns the first 50ms of the CIP period of cip_trace, t.

Usage:

```
the_period = periodPulseIni50ms(t)
```

Parameters:

`t`: A trace object.

Returns:

the_period: A period object.

See also: `period` (p. 140), `cip_trace` (p. 56), `trace` (p. 278)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/25

A.2.32 Method `cip_trace/plot_abstract`

Summary: Plots a trace by calling plotData.

Usage:

```
a_plot = plot_abstract(t, title_str, props)
```

Description: If t is a vector of traces, returns a vector of plot objects.

Parameters:

`t`: A trace object.

`title_str`: (Optional) String to append to plot title.

`props`: A structure with any optional properties.

`timeScale`: 's' for seconds, or 'ms' for milliseconds.
(rest passed to plot_abstract.)

Returns:

a_plot: A plot_abstract object that can be visualized.

See also: `trace` (p. 278), `trace/plot` (p. 280), `plot_abstract` (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/17

A.2.33 Method `cip_trace/getProfileAllSpikes`

Summary: Creates a cip_trace_allspikes_profile object by collecting test results of a cip_trace, analyzing each individual spike.

Usage:

```
profile_obj = getProfileAllSpikes(a_cip_trace)
```

Description: Analyzes the spontaneous (periodIniSpont), pulse (periodPulse) and the recovery (periodRecSpont) periods separately and produces spike shape distribution results. Rate and CIP measurements are added to these.

Parameters:

`a_cip_trace`: A cip_trace object.

Returns:

profile_obj: A cip_trace_allspikes_profile object.

See also: `cip_trace` (p. 56), `cip_trace_allspikes_profile` (p. 69)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/04/26

A.3 Class `cip_trace_allspikes_profile`

A.3.1 Constructor `cip_trace_allspikes_profile/cip_trace_allspikes_profile`

Summary: Creates and collects test results of a cip_trace.

Usage:

```
obj = cip_trace_allspikes_profile(a_cip_trace, a_spikes, a_spont_spike_shape, results, id,
props)
```

Description: This is a subclass of results_profile. It is made to be used from subclass constructors.

Parameters:

`a_cip_trace`: A cip_trace object.

`a_spikes`: A spikes object.

`spont_spikes_db, pulse_spikes_db, recov_spikes_db`: tests_dbs with spontaneous, pulse and recovery period spike info.

`results_obj`: A results_profile object with test results.

`id`: Identification string.

`props`: A structure with any optional properties.

Returns a structure object with the following fields:

trace, spikes, spont_spikes_db, pulse_spikes_db, recov_spikes_db, props

See also: `cip_trace` (p. 56), `spikes` (p. 202), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/05/04

A.3.2 Method `cip_trace_allspikes_profile/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.3.3 Method `cip_trace_allspikes_profile/set`

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.3.4 Method `cip_trace_allspikes_profile/plotRowSpontSpikeAnal`

Summary: Creates a row of plots that show spontaneous spikes, starting from the whole trace, zooming into the individual spike.

Usage:

```
a_plot = plotRowSpontSpikeAnal(prof, title_str)
```

Parameters:

`prof`: A cip_trace_allspikes_profile object.

`title_str`: (Optional) String to append to plot title.

Returns:

a_plot: A plot_abstract object that can be visualized.

See also: trace (p. 278), cip_trace (p. 56), spike_shape/plotCompareMethodsSimple (p. 194), plot_abstract (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/05/23

A.3.5 Method cip_trace_allspikes_profile/display

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.4 Class cip_trace_profile

A.4.1 Constructor cip_trace_profile/cip_trace_profile

Summary: Creates and collects test results of a cip_trace.

Description: The first usage is fully customizable to be used from subclass constructors. The second usage generates the spikes and spont_spike_shape objects, and collects some generic test results from them.

Parameters:

`data_src`**:** The trace column OR the filename.

`dt`**:** Time resolution [s]

`dy`**:** y-axis resolution [ISI (V, A, etc.)]

`pulse_time_start, pulse_time_width`**:** Start and width of the pulse [dt]

`id`**:** Identification string.

`props`**:** See trace object.

Returns a structure object with the following fields:

trace, spikes, spont_spike_shape, results, id, props.

See also: cip_trace (p. 56), spikes (p. 202), spike_shape (p. 189)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/25

A.4.2 Method cip_trace_profile/get

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.4.3 Method `cip_trace_profile/plot`

Summary: Plots a cip_trace_profile object.

Usage:

```
h = plot(t)
```

Description: Plots contents of this object.

Parameters:

`t`: A cip_trace_profile object.

Returns:

h: Plot handle(s).

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/15

A.4.4 Method `cip_trace_profile/set`

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.4.5 Method `cip_trace_profile/subsref`

Summary: Defines generic indexing for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.4.6 Method `cip_trace_profile/display`

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.5 Class `cip_traces_dataset`

A.5.1 Constructor `cip_traces_dataset/cip_traces_dataset`

Summary: Dataset of cip_traces objects, each with varying cip magnitudes.

Usage:

```
obj = cip_traces_dataset(ts, cipmag, id, props)
```

Description: This is a subclass of params_tests_fileset.

Parameters:

- `ts`: A cell array of cip_traces objects.
- `cipmag`: A single cip magnitude to trace take from objects.
- `id`: An identification string for the whole dataset.
- `props`: A structure with any optional properties passed to cip_trace_profile.

Returns a structure object with the following fields:

params_tests_dataset, cipmag, props (see above).

See also: `cip_traces` (p. **??**), `params_tests_fileset` (p. 135), `params_tests_db` (p. 118)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/30

A.5.2 Method `cip_traces_dataset/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.5.3 Method `cip_traces_dataset/set`

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.5.4 Method `cip_traces_dataset/getItemParams`

Usage:

```
params_row = getParams(dataset, index)
```

Parameters:

- `dataset`: A params_tests_dataset.
- `index`: Index of item in dataset.

Returns:

params_row: Parameter values in the same order of paramNames

See also: `itemResultsRow` (p. **??**), `params_tests_dataset` (p. 113), `paramNames` (p. **??**), `testNames` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/12/06

A.5.5 Method `cip_traces_dataset/cip_trace_profile`

Summary: Loads a raw cip_trace_profile given a index to this dataset.

Usage:

```
a_cip_trace_profile = cip_trace_profile(dataset, index)
```

Parameters:

`dataset`: A params_tests_dataset.

`index`: Index of file in dataset.

Returns:

a_cip_trace_profile: A cip_trace_profile object.

See also: `cip_trace_profile` (p. 71), `params_tests_dataset` (p. 113)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.5.6 Method `cip_traces_dataset/subsref`

Summary: Defines generic indexing for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.5.7 Method `cip_traces_dataset/display`

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.5.8 Method `cip_traces_dataset/loadItemProfile`

Summary: Loads a profile object from a raw data item in the dataset.

Usage:

```
a_profile = loadItemProfile(dataset, index)
```

Description: Subclasses should overload this function to load the specific profile object they desire. The profile class should define a getResults method which is used in the itemResultsRow method.

Parameters:

`dataset`: A params_tests_dataset.

`index`: Index of item in dataset.

Returns:

a_profile: A profile object that implements the getResults method.

See also: itemResultsRow (p. ??), params_tests_dataset (p. 113), paramNames (p. ??), testNames (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.5.9 Method cip_traces_dataset/paramNames

Summary: Returns the only parameter, 'pAcip,' for this fileset.

Usage:

```
param_names = paramNames(fileset)
```

Description: Looks at the filename of the first file to find the parameter names.

Parameters:

fileset: A params_tests_fileset.

Returns:

params_names: Cell array with ordered parameter names.

See also: params_tests_fileset (p. 135), paramNames (p. ??), testNames (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/12/06

A.6 Class cip_traceset

A.6.1 Constructor cip_traceset/cip_traceset

Summary: A traceset with varying cip magnitudes from a single cip_traces object.

Usage:

```
obj = cip_traceset(ct, cip_mags, dy, props)
```

Description: This is a subclass of params_tests_fileset. This traceset can create a mini-database form a single cip_traces object. The list contains cip_mags. cip_traceset_dataset should be used to load multiple cip_traceset objects.

Parameters:

ct: A cip_traces object.

cip_mags: An array of cip magnitudes to select from the object.

dy: y-axis resolution, [V] or [A] (default=1e-3).

`props`: A structure with any optional properties.

`offsetPotential`: Add this to physiology trace as compensation.

Returns a structure object with the following fields:

params_tests_dataset, ct, props (see above).

See also: `cip_traces` (p. **??**), `params_tests_fileset` (p. 135), `params_tests_db` (p. 118)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/30

A.6.2 Method `cip_traceset/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.6.3 Method `cip_traceset/getItemParams`

Usage:

```
params_row = getParams(dataset, index)
```

Parameters:

`dataset`: A params_tests_dataset.

`index`: Index of item in dataset.

`a_profile`: A profile object for the item (optional).

Returns:

params_row: Parameter values in the same order of paramNames

See also: `itemResultsRow` (p. **??**), `params_tests_dataset` (p. 113), `paramNames` (p. **??**), `testNames` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/12/06

A.6.4 Method `cip_traceset/cip_trace_profile`

Summary: Loads a raw cip_trace_profile given an index in this traceset.

Usage:

```
a_cip_trace_profile = cip_trace_profile(traceset, index)
```

Parameters:

`traceset`: A cip_traceset.

`index`: Index of item in traceset.

Returns:

a_cip_trace_profile: A cip_trace_profile object.

See also: `cip_trace_profile` (p. 71), `params_tests_dataset` (p. 113)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.6.5 Method `cip_traceset/display`

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.6.6 Method `cip_traceset/loadItemProfile`

Summary: Loads a profile object from a raw data item in the dataset.

Usage:

```
a_profile = loadItemProfile(dataset, index)
```

Description: Subclasses should overload this function to load the specific profile object they desire. The profile class should define a getResults method which is used in the itemResultsRow method.

Parameters:

`dataset`: A params_tests_dataset.

`index`: Index of item in dataset.

Returns:

a_profile: A profile object that implements the getResults method.

See also: `itemResultsRow` (p. **??**), `params_tests_dataset` (p. 113), `paramNames` (p. **??**), `testNames` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.6.7 Method `cip_traceset/paramNames`

Summary: Returns the only parameter, 'pAcip,' for this traceset.

Usage:

```
param_names = paramNames(traceset)
```

Description: Looks at the filename of the first file to find the parameter names.

Parameters:

`traceset`**:** A cip_traceset.

Returns:

params_names: Cell array with ordered parameter names.

See also: `params_tests_dataset` (p. 113), `paramNames` (p. **??**), `testNames` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/12/06

A.7 Class `cip_traceset_dataset`

A.7.1 Constructor `cip_traceset_dataset/cip_traceset_dataset`

Summary: Dataset of multiple cip magnitudes from cip_traces objects .

Usage:

```
obj = cip_traceset_dataset(cts, cip_mags, dy, id, props)
```

Description: This is a subclass of params_tests_dataset. Designed to extract a trace for each cip magnitude from the cip_traceset objects contained. Uses cip_traceset objects to extract multiple traces from each cip_traces object.

Parameters:

`cts`**:** Array or cell array of cip_traces objects.

`cip_mags`**:** An array of cip magnitudes to select from each cip_traces object.

`dy`**:** y-axis resolution, [V] or [A] (default = 1e-3).

`id`**:** An identification string.

`props`**:** A structure with any optional properties passed to cip_traceset.

Returns a structure object with the following fields:

params_tests_dataset, cip_mags

See also: `physiol_cip_traceset` (p. 147), `params_tests_dataset` (p. 113), `params_tests_db` (p. 118)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/01/28

A.7.2 Method `cip_traceset_dataset/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.7.3 Method `cip_traceset_dataset/readDBItems`

Summary: Reads all items to generate a params_tests_db object.

Usage:

```
[params, param_names, tests, test_names] = readDBItems(obj)
```

Description: This is a specific method to convert from cip_traceset_dataset to a params_tests_db, or a subclass. Output of this function can be directly fed to the constructor of a params_tests_db or a subclass.

Parameters:

`obj`: A physiol_cip_traceset_fileset

Returns:

params, param_names, tests, test_names: See params_tests_db.

See also: `params_tests_db` (p. 118), `params_tests_fileset` (p. 135), `itemResultsRow` `testNames` (p. **??**), `paramNames` (p. **??**), `physiol_cip_traceset_fileset` (p. 152)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/01/28

A.7.4 Method `cip_traceset_dataset/display`

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.7.5 Method `cip_traceset_dataset/loadItemProfile`

Summary: Loads a cip_trace_profile object from a raw data file in the fileset.

Usage:

```
a_profile = loadItemProfile(fileset, neuron_id, trace_index)
```

Parameters:

`fileset`: A physiol_cip_traceset object.

`neuron_id` **:** tells which item in fileset (corresponds to cells_filename) to use grab the cell information

`trace_index`**:** Index of file in traceset.

Returns:

a_profile: A profile object that implements the getResults method.

See also: `itemResultsRow` (p. **??**), `params_tests_fileset` (p. 135), `paramNames` (p. **??**), `testNames` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14 and Tom Sangrey

A.8 Class `cluster_db`

A.8.1 Constructor `cluster_db/cluster_db`

Summary: A database of cluster centroids generated by a clustering algorithm from a rows of orig_db.

Usage:

```
a_cluster_db = cluster_db(data, col_names, orig_db, cluster_idx, id, props)
```

Description: This is a subclass of tests_db. Use one of the clustering methods in tests_db, such as kmeansCluster, to get an instance of this class.

Parameters:

`data`**:** Database contents.

`col_names`**:** The column names.

`orig_db`**:** DB whose rows are clustered.

`cluster_idx`**:** Array of cluster numbers that correspond to each row in orig_db.

`id`**:** An identifying string.

`props`**:** A structure with any optional properties.

`sumDistances`**:** Total distance of elements within each cluster.

`distanceMeasure`**:** Measure used to find clusters (Default='correlation')

Returns a structure object with the following fields:

tests_db, orig_db: original DB from which clusters were obtained, cluster_idx: Array associating rows of orig_db to each cluster here. props.

See also: `tests_db` (p. 227), `tests_db/kmeansCluster` (p. 258)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/04/08

A.8.2 Method `cluster_db/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.8.3 Method `cluster_db/plotHist`

Summary: Creates a histogram plot showing the clustering memberships.

Usage:

```
a_plot = plotHist(a_cluster_db, title_str)
```

Parameters:

`a_cluster_db`**:** A cluster_db object.

`title_str`**:** (Optional) String to append to plot title.

Returns:

a_plot: A plot_abstract object that can be plotted.

See also: `plot_abstract` (p. 157), `plotFigure` (p. **??**), `histogram_db` (p. 96), `histogram_db/plot_abstract` (p. 99)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/04/08

A.8.4 Method `cluster_db/plotQuality`

Summary: Creates a plot_abstract of the silhouette plot showing the clustering quality.

Usage:

```
a_plot = plotQuality(a_cluster_db, title_str)
```

Parameters:

`a_cluster_db`**:** A cluster_db object.

`title_str`**:** (Optional) String to append to plot title.

Returns:

a_plot: A plot_abstract object that can be plotted.

See also: `plot_abstract` (p. 157), `plotFigure` (p. **??**), `silhouette` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/04/08

A.8.5 Method cluster_db/display

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.8.6 Method cluster_db/plot_abstract

Summary: Creates a vertical plot_stack of silhouette and membership histograms for the clusters.

Usage:

```
a_plot = plot_abstract(a_cluster_db, title_str)
```

Parameters:

`a_cluster_db`: A cluster_db object.

`title_str`: (Optional) String to append to plot title.

Returns:

a_plot: A plot_abstract object that can be plotted.

See also: `cluster_db/plotQuality` (p. 81), `cluster_db/plotHist` (p. 81)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/04/08

A.9 Class corrcoefs_db

A.9.1 Constructor corrcoefs_db/corrcoefs_db

Summary: A database of correlation coefficients generated from a column of another database.

Usage:

```
a_coef_db = corrcoefs_db(col_name, coefs, coef_names, pages, id, props)
```

Description: This is a subclass of tests_3d_db. Allows generating a plot, etc.

Parameters:

`col_name`: The column with which the others are correlated.

`coefs`: Matrix where each column has another coefficient.

`coef_names`: Cell array of column names corresponding to coefficients.

`pages`: Column vector of page indices pointing to the tests_3d_db.

`id`: An identifying string.

`props`: A structure with any optional properties.

Returns a structure object with the following fields:

tests_db.

See also: `tests_db` (p. 227), `plot_simple` (p. 169), `tests_db/histogram` (p. 237)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/06

A.10 Class `dataset_db_bundle`

A.10.1 Constructor `dataset_db_bundle/dataset_db_bundle`

Summary: The dataset and the DB created from it bundled together.

Usage:

```
a_bundle = dataset_db_bundle(a_dataset, a_db, a_joined_db, props)
```

Description: This class is made to enable operations that require seamless connection between the high-level DB and the raw data. The raw DB is only required to bridge the gap between the high-level DB and the dataset. Therefore it only needs to contain columns necessary to make this connection. It is not required to include all raw DB columns, which is inefficient.

Parameters:

`a_dataset`**:** A params_tests_dataset object or a subclass.

`a_db`**:** The raw tests_db object (or a subclass) created from the dataset.

`a_joined_db`**:** The processed DB created from the raw DB.

`props`**:** A structure with any optional properties.

Returns a structure object with the following fields:

dataset, db, joined_db, props.

See also: `tests_db` (p. 227), `params_tests_dataset` (p. 113)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/12/13

A.10.2 Method `dataset_db_bundle/matchingRow`

Summary: Creates a criterion database for matching the tests of a row.

Usage:

```
crit_db = matchingRow(a_bundle, row, props)
```

Description: Copies selected test values from row as the first row into the criterion db. Adds a second row for the STD of each column in the db.

Parameters:

- `a_bundle`**:** A tests_db object.
- `row`**:** A row index to match.
- `props`**:** A structure with any optional properties.
  - `distDB`**:** Take the standard deviation from this db instead.

Returns:

crit_db: A tests_db with two rows for values and STDs.

Example:

```
physiol_bundle has an overloaded matchingRow method that
takes the TracesetIndex as argument:
» crit_db = matchingRow(pbundle, 61)
```

See also: `rankMatching` (p. **??**), `tests_db` (p. 227), `tests2cols` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/12/21

A.10.3 Method `dataset_db_bundle/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.10.4 Method `dataset_db_bundle/getNeuronRowIndex`

Summary: Returns the neuron index from bundle.

Usage:

```
a_row_index = getNeuronRowIndex(a_bundle, an_index, props)
```

Description: This is a polymorphic method. Therefor it is not defined for this class, but see subclasses of dataset_db_bundle for its more meaningful implementations.

Parameters:

- `a_bundle`**:** A dataset_db_bundle subclass object.
- `an_index`**:** An index number of neuron, or a DB row containing this.
- `props`**:** A structure with any optional properties.

Returns:

a_row_index: A row index of neuron in a_bundle.joined_db.

Example:

```
» displayRows(mbundle.joined_db(getNeuronRowIndex(mbundle, 98364), :))
```

See also: `dataset_db_bundle` (p. 83)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/06/09

A.10.5 Method `dataset_db_bundle/ctFromRows`

Summary: Loads a cip_trace object from a raw data file in the a_bundle.

Usage:

```
a_cip_trace = ctFromRows(a_bundle, a_db, cip_levels, props)
```

Description: This method is not implemented for the generic dataset_db_bundle class. See subclass implementations.

Parameters:

`a_bundle`**:** A dataset_db_bundle object.

`a_db`**:** A DB created by the dataset in the a_bundle to read the neuron index numbers from.

`cip_levels`**:** A column vector of CIP-levels to be loaded.

`props`**:** A structure with any optional properties.
(passed to a_bundle.dataset/cip_trace)

Returns:

a_cip_trace: One or more cip_trace objects that hold the raw data.

See also: `model_ct_bundle/ctFromRows` (p. 102), `physiol_bundle/ctFromRows` (p. 144)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/10/11

A.10.6 Method `dataset_db_bundle/reportNeuron`

Summary: Generates a report of neuron at given an_index of a_bundle.

Usage:

```
a_doc = reportNeuron(a_bundle, an_index, props)
```

Description: Generates a report document with preset layouts of annotated plots of the selected neuron. See reportLayout below for presets.

Parameters:

`a_bundle`**:** a dataset_db_bundle object which contains the neuron

`an_index`**:** The index to pass to ctFromRows method of a_bundle.

`props`**:** A structure with any optional properties.

- `reportLayout`**:** Allows choosing one of predefined report types (strings):
  - `1`**:** Only +/- 100 pA traces in one plot (default).
    1a/b: Either one of the +/- 100 pA traces in one plot.
  - `2`**:** Only +/- 100 pA traces and spike shapes in one horiz. plot.
  - `2a`**:** +/- 100 pA traces, f-I curve and spike shapes in one horiz. plot.
  - `3`**:** +100 pA raw trace and rate profile stacked vertically.
  - `3b`**:** -100 pA raw trace and rate profile stacked vertically.
  - `4`**:** Horiz stack of +/- 100 pA raw trace with rate profiles underneath.
  - `5`**:** 5-piece trace, spike shape, f-I curve, f-t curve quad-plot.
- `numTraces`**:** Limit number of traces to show in plot (>=1).
- `traces`**:** List of acceptable traces to load.
- `traceAxisLimits`**:** If given, use these limits for trace plots.
- `rateAxisLimits`**:** If given, use these limits for rate plots.
- `fIAxisLimits`**:** If given, use these limits for fIcurve plots.
- `fIstats`**:** Add a fI-stats plot in addition to the curve.
- `sshapeAxisLimits`**:** If given, use these limits for spike shape plots.
- `sshapeResults`**:** If 1, plot measures on the spike shape (default=1).

Returns:

a_doc: A doc_generate object, or a subclass, that can be plotted, or printed as a PS or PDF file.

Example:

```
» printTeXFile(reportNeuron(mbundle, 2222), 'a.tex')
or:
» plotFigure(get(reportNeuron(mbundle, 2222), 'plot'))
or if the result is one or many doc_plot objects:
» plot(reportNeuron(mbundle, 2222:2224, struct('reportLayout', '5')))
```

See also: `doc_multi` (p. 92), `doc_generate` (p. 89), `doc_generate/printTeXFile` (p. 90)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/01/24

A.10.7 Method `dataset_db_bundle/set`

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.10.8 Method dataset_db_bundle/constrainedMeasuresPreset

Summary: Returns a dataset_db_bundle with constrained measures according to chosen preset.

Usage:

```
[a_bundle test_names] = constrainedMeasuresPreset(a_bundle, preset, props)
```

Parameters:

`a_bundle`**:** A dataset_db_bundle object.

`preset`**:** Choose preset measure list (default=1).

`props`**:** A structure with any optional properties.

Returns:

a_bundle: Modified bundle.

See also: `physiol_bundle/constrainedMeasuresPreset` (p. 146)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/06/13

A.10.9 Method dataset_db_bundle/subsref

Summary: Defines indexing for tests_db objects for () and . operations.

Usage:

```
obj = obj(rows, tests) obj = obj.attribute
```

Description: Returns attributes or selects the given test columns and rows and returns in a new tests_db object.

Parameters:

`obj`**:** A tests_db object.

`rows`**:** A logical or index vector of rows. If ':', all rows.

`tests`**:** Cell array of test names or column indices. If ':', all tests.

`attribute`**:** A tests_db class attribute.

Returns:

obj: The new tests_db object.

See also: `subsref` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/17

A.10.10 Method dataset_db_bundle/subsasgn

Summary: Defines generic index-based assignment for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/02/06

A.10.11 Method dataset_db_bundle/plotfICurve

Summary: Generates a f-I curve doc_plot for neuron at given an_index in a_bundle.

Usage:

```
a_plot = plotfICurve(a_bundle, trial_num, props)
```

Parameters:

`a_bundle`**:** A dataset_db_bundle object.

`an_index`**:** An index with which to address the a_bundle.

`props`**:** A structure with any optional properties.

- `shortCaption`**:** This appears in the figure caption.
- `plotMStats`**:** If set, add the a_bundle stats plot.
- `captionToStats`**:** Use this as its legend label.
- `quiet`**:** if given, no title is produced
  (passed to plot_superpose)

Returns:

a_plot: A plot_superpose that contains a f-I curve plot.

Example:

```
» a_p = plotfICurve(r, 1);
» plotFigure(a_p, 'The f-I curve of best matching model');
```

See also: `plot_abstract` (p. 157), `plot_superpose` (p. 173)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/01/16

A.10.12 Method dataset_db_bundle/display

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.10.13 Method `dataset_db_bundle/rankingReportTeX`

Summary: Generates a report by comparing a_bundle with the given match criteria, crit_db from crit_bundle.

Usage:

```
tex_string = rankingReportTeX(a_bundle, crit_bundle, crit_db, props)
```

Description: Generates a LaTeX document with: - (optional) Raw traces compared with some best matches at different distances - Values of some top matching a_db rows and match errors in a floating table. - colored-plot of measure errors for some top matches. - Parameter distributions of 50 best matches as a bar graph.

Parameters:

`a_bundle`: A dataset_db_bundle object that contains the DB to compare rows from.

`crit_bundle`: A dataset_db_bundle object that contains the criterion dataset.

`crit_db`: A tests_db object holding the match criterion tests and STDs which can be created with matchingRow.

`props`: A structure with any optional properties.

`caption`: Identification of the criterion db (not needed/used?).

`num_matches`: Number of best matches to display (default=10).

`rotate`: Rotation angle for best matches table (default=90).

Returns:

tex_string: LaTeX document string.

See also: `displayRowsTeX` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/12/13

A.11 Class `doc_generate`

A.11.1 Constructor `doc_generate/doc_generate`

Summary: Generic class to help generate printed or annotated documents with results.

Usage:

```
a_doc = doc_generate(text_string, id, props)
```

Description: This constitutes the base class for other doc_ classes. For convenience, this class holds a text_string to be printed when the document is generated with the printTeXFile option.

Parameters:

`text_string`**:** Contents of this document.

`id`**:** An identifying string.

`props`**:** A structure with any optional properties.

Returns a structure object with the following fields:

text, id, props.

See also: `doc_plot` (p. 93), `doc_multi` (p. 92)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/01/17

A.11.2 Method `doc_generate/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.11.3 Method `doc_generate/set`

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.11.4 Method `doc_generate/printTeXFile`

Summary: Creates a TeX file with the contents of this document.

Usage:

```
printTeXFile(a_doc, filename, props)
```

Description: Calls getTeXString to generate the contents. The filename is adjusted with a call to properFilename to generate an acceptable TeX filename. TeX-specific should only be added at this point or at getTeXString, because before we want the object to be a generic document container.

Parameters:

`a_doc`**:** A tests_db object.

`filename`**:** To write the TeX string.

`props`**:** A structure with any optional properties.

Returns:

tex_string: A string that contains TeX commands, which upon writing to a file, can be interpreted by the TeX engine to produce a document.

Example:

```
» a_doc = doc_plot(a_plot, 'Results from cell.', 'Results.', struct, '');
» printTeXFile(a_doc, 'my_doc.tex')
then my_doc.tex can be used by including from a valid LaTeX document.
```

See also: `doc_generate` (p. 89), `doc_plot` (p. 93), `string2File` (p. **??**), `properFilename` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/01/17

A.11.5 Method `doc_generate/subsref`

Summary: Defines generic indexing for objects.

A.11.6 Method `doc_generate/display`

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.11.7 Method `doc_generate/getTeXString`

Summary: Returns the TeX representation for the document.

Usage:

```
tex_string = getTeXString(a_doc, props)
```

Description: This is an abstract placeholder for this method. It specifies what this method should do in the subclasses that implement it. This method should create all the auxiliary files needed by the document. The generated tex_string should be ready to be visualized.

Parameters:

`a_doc`: A tests_db object.

`props`: A structure with any optional properties.

Returns:

tex_string: A string that contains TeX commands, which upon writing to a file, can be interpreted by the TeX engine to produce a document.

Example:

```
doc_plot has an overloaded getTeXString method:
» tex_string = getTeXString(a_doc_plot)
» string2File(tex_string, 'my_doc.tex')
then my_doc.tex can be used by including from a valid LaTeX document.
```

See also: `doc_generate` (p. 89), `doc_plot` (p. 93)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/01/17

A.12 Class `doc_multi`

A.12.1 Constructor `doc_multi/doc_multi`

Summary: A document that is composed of multiple other doc_generate objects.

Usage:

```
a_doc = doc_multi(docs, id, props)
```

Parameters:

`docs`**:** A vector of doc_generate objects.

`id`**:** An identifying string.

`props`**:** A structure with any optional properties.

Returns a structure object with the following fields:

docs, doc_generate.

Example:

```
» mydoc = doc_multi([doc_plot(a_plot1), doc_plot(a_plot2)], 'Two plots')
» printTeXFile(mydoc, 'two_plots.tex')
```

See also: `doc_generate` (p. 89), `getTeXString` (p. **??**), `doc_generate/printTeXFile` (p. 90)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/01/17

A.12.2 Method `doc_multi/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.12.3 Method doc_multi/set

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.12.4 Method doc_multi/getTeXString

Summary: Returns the TeX representation for the document.

Usage:

```
tex_string = getTeXString(a_doc, props)
```

Description: Concatenates TeX representations of doc_generate, or subclass, objects it contains.

Parameters:

a_doc: A tests_db object.

props: A structure with any optional properties.

Returns:

tex_string: A string that contains TeX commands, which upon writing to a file, can be interpreted by the TeX engine to produce a document.

Example:

```
doc_plot has an overloaded getTeXString method:
» tex_string = getTeXString(a_doc_plot)
» string2File(tex_string, 'my_doc.tex')
then my_doc.tex can be used by including from a valid LaTeX document.
```

See also: doc_generate (p. 89), doc_plot (p. 93)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/01/17

A.13 Class doc_plot

A.13.1 Constructor doc_plot/doc_plot

Summary: Generates a formatted plot for printing, annotated with captions.

Usage:

```
a_doc = doc_plot(a_plot, caption, plot_filename, float_props, id, props)
```

Description: The generated file may take an extension according to chosen format.

Parameters:

`a_plot`: A plot_abstract ready to be visualized.

`caption`: Long caption to appear under the figure.

`plot_filename`: Filename to be generated from the plot.

`float_props`: Formatting instructions passed to TeXtable.

`id`: An identifying string.

`props`: A structure with any optional properties.

  `orient`: Passed to the orient command before printing to figure file.

Returns a structure object with the following fields:

plot, caption, plot_filename, float_props, doc_generate.

Example:

```
» a_doc = doc_plot(plotData(my_cip_trace), 'My CIP trace.  Very interesting.', ...
'trace1', struct, 'first doc');
» printTeXFile(a_doc, 'my_doc.tex');
```

See also: `doc_generate` (p. 89), `TeXtable` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/01/17

A.13.2 Method `doc_plot/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.13.3 Method `doc_plot/plot`

Summary: Default plot method to preview the contained plot in a new figure.

Usage:

```
figure_handle = plot(a_doc, props)
```

Description: Only generate the contained plot for previewing. Opens a new figure.

Parameters:

`a_doc`: A doc_plot object.

`props`: A structure with any optional properties.

Returns:

figure_handle: Handle of newly opened figure.

Example:

```
» figure_handle = plot(a_doc_plot)
```

See also: plot_abstract/plotFigure (p. 162), doc_generate (p. 89), doc_plot (p. 93)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/01/17

A.13.4 Method doc_plot/set

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.13.5 Method doc_plot/plot_abstract

Summary: Returns the plot_abstract object within the doc_plot.

Usage:

```
a_plot = plot_abstract(a_doc, title_str, props)
```

Description: If a_doc is a vector, returns a vector of plot_abstract objects.

Parameters:

- a_doc: A doc_plot object.
- title_str: (Optional) String to replace plot title.
- props: A structure with any optional properties.
  (rest passed to plot_abstract.)

Returns:

a_plot: A plot_abstract object or vector that can be visualized.

See also: doc_plot/plot (p. 94), plot_abstract (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/17

A.13.6 Method doc_plot/getTeXString

Summary: Returns the TeX representation for the plot document.

Usage:

```
tex_string = getTeXString(a_doc, props)
```

Description: Plots, prints EPS files and generates the necessary LaTeX code.

Parameters:

`a_doc`**:** A doc_plot object.

`props`**:** A structure with any optional properties.

Returns:

tex_string: A string that contains TeX commands, which upon writing to a file, can be interpreted by the TeX engine to produce a document.

Example:

```
doc_plot has an overloaded getTeXString method:
» tex_string = getTeXString(a_doc_plot)
» string2File(tex_string, 'my_doc.tex')
then my_doc.tex can be used by including from a valid LaTeX document.
```

See also: `doc_generate` (p. 89), `doc_plot` (p. 93)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/01/17

A.14 Class `histogram_db`

A.14.1 Constructor `histogram_db/histogram_db`

Summary: A database of histogram values generated for a column of another database.

Usage:

```
a_hist_db = histogram_db(col_name, bins, hist_results, id, props)
```

Description: This is a subclass of tests_db. Allows generating a histogram plot, etc. The histogram count is entered as a column named histVal.

Parameters:

`col_name`**:** The column name of the histogrammed value.

`bins`**:** The values for which the histogram values are calculated.

`hist_results`**:** A column vector of histogram values.

`id`**:** An identifying string.

`props`**:** A structure with any optional properties.

Returns a structure object with the following fields:

tests_db, props.

Example:

```
» [hist_results, bins] = hist(my_data);
» a_hist_db = histogram_db('firing_rate', bins, hist_results, 'rate histogram db');
» plot(a_hist_db);
```

See also: tests_db (p. 227), plot_simple (p. 169), tests_db/histogram (p. 237)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/20

A.14.2 Method histogram_db/plotEqSpaced

Summary: Generates a histogram plot where the values are equally spaced on the x-axis. For use with non-linear parameter values.

Usage:

```
a_plot = plotEqSpaced(a_hist_db, title_str, props)
```

Description: Generates a plot_simple object from this histogram.

Parameters:

- a_hist_db: A histogram_db object.
- title_str: Optional title string.
- props: Optional properties passed to plot_abstract.
  - quiet: If 1, don't include database name on title.

Returns:

a_plot: A object of plot_abstract or one of its subclasses.

See also: plot_abstract (p. 157), plot_simple (p. 169)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/22

A.14.3 Method histogram_db/get

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.14.4 Method histogram_db/calcMode

Summary: Finds the mode of the distribution, that is, the bin with the highest value.

Usage:

```
[mode_val, mode_mag] = calcMode(a_hist_db)
```

Parameters:

- a_hist_db: A histogram_db object.

Returns:

mode_val: The center of the bin that has most members. mode_mag: The value of the histogram bin.

See also: histogram_db (p. 96)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/04/27

A.14.5 Method histogram_db/plotRowMatrix

Summary: Generates a subplot matrix of measure columns versus rows of databases.

Usage:

```
a_plot = plotRowMatrix(hist_dbs, title_str, props)
```

Description: Each row in the hist_dbs is assumed to come from a different DB. Columns represent histograms of different measurements. The plot is made such that histograms in each row have the same maximal count, and histograms in each column have the same axis limits.

Parameters:

`hist_dbs`**:** A matrix of histogram_db objects.

`title_str`**:** Title to go at the top.

`props`**:** A structure with any optional properties.

> `rowLabels`**:** Cell array of y-axis labels for each row.
> (rest passed to histogram_db/plot_abstract)

Returns:

a_plot: A object of plot_abstract or one of its subclasses.

See also: plot_abstract (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/11/22

A.14.6 Method histogram_db/subsref

Summary: Defines generic indexing for objects.

A.14.7 Method histogram_db/plotPages

Summary: Generates a plot containing subplots from each page of histograms.

Usage:

```
a_plot = plotPages(a_hist_db, title_str, props)
```

Description: For each page of the histogram, a histogram is placed in a subplot.

Parameters:

a_hist_db: A histogram_db object.
title_str: (Optional) String to append to plot title.
props: A structure with any optional properties, passed to plot_stack.
an_orient: Stack orientation. One of 'x', or 'y' (Default='y'; vertical).
quiet: If 1, only display given title_str.

Returns:

a_plot: A object of plot_abstract or one of its subclasses.

See also: plotPages (p. **??**), plot_simple (p. 169)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/04

A.14.8 Method histogram_db/plot_abstract

Summary: Creates a bar plot from the histogram.

Usage:

```
a_plot = plot_abstract(a_hist_db, title_str, props)
```

Description: Generates a plot_simple object from this histogram.

Parameters:

a_hist_db: A histogram_db object.
props: Optional properties passed to plot_abstract.
command: Plot command (Optional, default='bar')
lineSpec: Line specification passed to bar command.
logScale: If 1, use logarithmic y-scale.
shading: 'faceted' (default) or 'flat'.
quiet: If 1, don't include database name on title.

Returns:

a_plot: A object of plot_abstract or one of its subclasses.

See also: plot_abstract (p. 157), plot_simple (p. 169)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/22

A.15 Class model_ct_bundle

A.15.1 Constructor model_ct_bundle/model_ct_bundle

Summary: The model cip_trace dataset and the DB created from it bundled together.

Usage:

```
a_bundle = model_ct_bundle(a_dataset, a_db, a_joined_db, props)
```

Description: This is a subclass of dataset_db_bundle, specialized for model datasets.

Parameters:

a_dataset: A params_cip_trace_fileset object.

a_db: The raw params_tests_db object created from the dataset. It only needs to have the pAcip, trial, and ItemIndex columns.

a_joined_db: The one-model-per-line DB created from the raw DB.

props: A structure with any optional properties.

Returns a structure object with the following fields:

dataset_db_bundle.

Example:

```
» a_joined_db = mergeMultipleCIPsInOne(a_db, ...)
» a_bundle = model_ct_bundle(a_params_cip_trace_fileset, a_db, a_joined_db)
```

See also: dataset_db_bundle (p. 83), tests_db (p. 227), params_tests_dataset (p. 113), params_tests_db/mergeMultipleCIPsInOne (p. 128)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/12/13

A.15.2 Method model_ct_bundle/reportRankingToPhysiolNeuronsTeXFile

Summary: Compare model DB to given physiol criterion and create a report.

Usage:

```
tex_filename = reportRankingToPhysiolNeuronsTeXFile(m_bundle, p_bundle, a_crit_db, props)
```

Description: A LaTeX report is generated following the example in physiol_bundle/matchingRow. The filename contains the neuron name, followed by the traceset index as an identifier of pharmacological applications, as in gpd0421c_s34.

Parameters:

m_bundle: A model_ct_bundle object.

p_bundle: A physiol_bundle object.

`a_crit_db`: The criterion neuron chosen with a matchingRow method.

`props`: A structure with any optional properties.

`filenameSuffix`: Append this identifier to the TeX filename.
(others passed to rankMatching)

Returns:

tex_filename: Name of LaTeX file generated.

See also: `tests_db/rankMatching` (p. 249), `physiol_cip_traceset/cip_trace` (p. 150), `physiol_bundle/matchingRow` (p. 143)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/01/18

A.15.3 Method `model_ct_bundle/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.15.4 Method `model_ct_bundle/plotComparefICurve`

Summary: Generates a f-I curve doc_plot comparing m_trial and to_index.

Usage:

```
a_plot = plotComparefICurve(m_bundle, m_trial, to_bundle, to_index, props)
```

Description: Note that this is not a general method. to_bundle should have been able to accept any type of bundle. Most probably this method is redundant and deprecated.

Parameters:

`m_bundle`: A model_ct_bundle object.

`m_trial`: Trial number of model.

`to_bundle`: A physiol_bundle object.

`to_index`: TracesetIndex of neuron.

`props`: A structure with any optional properties.

`shortCaption`: This appears in the figure caption.
`plotMStats`: If set, add the m_bundle stats plot.
`plotToStats`: If set, add the to_bundle stats plot.
`captionToStats`: Use this as its legend label.
`quiet`: if given, no title is produced
(passed to plot_superpose)

Returns:

a_plot: A plot_superpose that contains a f-I curve plot.

Example:

```
» a_p = plotComparefICurve(r, 1);
» plotFigure(a_p, 'The f-I curve of best matching model');
```

See also: plot_abstract (p. 157), plot_superpose (p. 173)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/01/16

A.15.5 Method model_ct_bundle/getNeuronRowIndex

Summary: Returns the neuron index from bundle.

Usage:

```
a_row_index = getNeuronRowIndex(a_bundle, trial_num, props)
```

Parameters:

a_bundle: A model_ct_bundle object.

trial_num: The trial number of model neuron, or a DB row containing this.

props: A structure with any optional properties.

Returns:

a_row_index: A row index of neuron in a_bundle.joined_db.

See also: dataset_db_bundle (p. 83)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/06/09

A.15.6 Method model_ct_bundle/ctFromRows

Summary: Loads a cip_trace object from a raw data file in the a_mbundle.

Usage:

```
a_cip_trace = ctFromRows(a_mbundle, a_db|trials, cip_levels, props)
```

Description: This is an overloaded method.

Parameters:

a_mbundle: A model_ct_bundle object.

a_db: A DB created by the dataset in the a_mbundle to read the trial numbers from.

`trials`: A column vector with trial numbers.

`cip_levels`: A column vector of CIP-levels to be loaded.

`props`: A structure with any optional properties.
(passed to a_mbundle.dataset/cip_trace)

Returns:

a_cip_trace: One or more cip_trace objects that hold the raw data.

See also: `dataset_db_bundle/ctFromRows` (p. 85)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/07/13

A.15.7 Method `model_ct_bundle/getNeuronLabel`

Summary: Constructs the neuron label from bundle.

Usage:

```
a_label = getNeuronLabel(a_bundle, trial_num, props)
```

Parameters:

`a_bundle`: A physiol_cip_traceset_fileset object.

`trial_num`: The trial number of model neuron.

`props`: A structure with any optional properties.

Returns:

a_label: A string label identifying selected neuron in bundle.

See also: `dataset_db_bundle` (p. 83)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/05/26

A.15.8 Method `model_ct_bundle/addToDB`

Summary: Concatenate to existing DB in the bundle.

Usage:

```
a_mbundle = addToDB(a_mbundle, a_raw_db, props)
```

Description: If joinedDb is not given in props, calls treatSimDB to get the joined_db from this raw DB. Then concats to both db and joined_db in bundle.

Parameters:

`a_mbundle`: A model_ct_bundle object.

`a_crit_bundle`**:** A physiol_bundle having a crit_db as its joined_db.

`props`**:** A structure with any optional properties.

`joinedDb`**:** The joined version of a_raw_db.

`dataset`**:** If given, this one is used to replace the fileset in the bundle.

Returns:

a_mbundle: a model_ct_bundle object containing the added DB.

Example:

```
» mbundle = addToDB(mbundle, params_tests_db(mfileset, [19684:59956]))
```

See also: `params_tests_fileset/addFiles` (p. 137), `multi_fileset_gpsim_cns2005/addFileDir` (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/02/06

A.15.9 Method `model_ct_bundle/set`

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.15.10 Method `model_ct_bundle/reportCompareModelToPhysiolNeuron`

Summary: Generates a report by comparing given model neuron to given physiol neuron.

Usage:

```
a_doc_multi = reportCompareModelToPhysiolNeuron(m_bundle, trial_num, p_bundle,
traceset_index, props)
```

Description: Generates a report document with: - Figure displaying raw traces of the physiol neuron compared with the model neuron - Figure comparing f-I curves of the two neurons. - Figure comparing spont and pulse spike shapes of the two neurons.

Parameters:

`m_bundle, p_bundle`**:** dataset_db_bundle objects of the model and physiology neurons.

`trial_num`**:** Trial number of desired model neuron in m_bundle.

`traceset_index`**:** TracesetIndex of desired neuron in p_bundle.

`props`**:** A structure with any optional properties.

`horizRow`: If defined, create a row-figure with all plots.

`numPhysTraces`: Number of physiology traces to show in plot (>=1).

Returns:

a_doc_multi: A doc_multi object that can be printed as a PS or PDF file.

Example:

```
» printTeXFile(reportCompareModelToPhysiolNeuron(mbundle, 2222, pbundle, 34), 'a.tex')
```

See also: `doc_multi` (p. 92), `doc_generate` (p. 89), `doc_generate/printTeXFile` (p. 90)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/01/24

A.15.11 Method `model_ct_bundle/rankMatching`

Summary: Create a ranked_db from given criterion db.

Usage:

```
a_ranked_db = rankMatching(a_mbundle, a_crit_db, props)
```

Parameters:

`a_mbundle`: A model_ct_bundle object.

`a_crit_db`: A crit_db created by a matchingRow method.

`props`: A structure with any optional properties.
(passed to tests_db/rankMatching)

Returns:

a_ranked_db: a ranked_db object containing the rankings.

See also: `tests_db/rankMatching` (p. 249), `ranked_db` (p. 176)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/01/18

A.15.12 Method `model_ct_bundle/collectPhysiolMatches`

Summary: Compare model DB to given physiol criteria and return some top matches.

Usage:

```
row_index = collectPhysiolMatches(a_mbundle, a_crit_bundle, props)
```

Parameters:

`a_mbundle`: A model_ct_bundle object.

`a_crit_bundle`: A physiol_bundle object that holds the criterion neuron.

`props`: A structure with any optional properties.

`showTopmost`: Number of top matching models to return (default=50)

Returns:

row_index: Row indices of best matching models.

See also:

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/01/18

A.15.13 Method `model_ct_bundle/getTrialNum`

Summary: Extracts identifying neuron trial number from DB.

Usage:

```
trial_num = getTrialNum(a_bundle, a_db|trial_num, props)
```

Parameters:

`a_bundle`: A physiol_cip_traceset_fileset object.

`a_db`: DB rows representing deisred model neuron(s).

`trial_num`: Trial numbers. If specified, this function does nothing but return them.

`props`: A structure with any optional properties.

Returns:

trial_num: The trial number(s) identifying selected neuron(s) in bundle.

See also: `dataset_db_bundle` (p. 83)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/05/26

A.15.14 Method `model_ct_bundle/plotCompareRanks`

Summary: Generates a plots of given ranks from the ranked_bundle.

Usage:

```
plots = plotCompareRanks(m_bundle, p_bundle, a_ranked_db, ranks, props)
```

Parameters:

`m_bundle`: A model_ct_bundle object.

`p_bundle`: A dataset_db_bundle object that originated the criterion.

`a_ranked_db`: A ranked_db generated from ranking m_bundle.

`ranks`: Vector of rank indices for which to generate the plots.

`props`: A structure with any optional properties.

Returns:

plots: A structure that contains the joined_db, and the plot vectors trace_d100_plots and trace_h100_plots.

Example:

```
» plots = plotCompareRanks(r, 1:10);
» plotFigure(plots.trace_d100_plots(1), 'The best matching +100 pA CIP trace');
```

See also:

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/01/16

A.16 Class `model_ranked_to_physiol_bundle`

A.16.1 Constructor `model_ranked_to_physiol_bundle/model_ranked_to_physiol_bundle`

Summary: A DB bundled with its dataset, ranked to a physiology DB bundle.

Usage:

```
r_bundle = model_ranked_to_physiol_bundle(a_dataset, a_db, a_ranked_db, a_crit_bundle, props)
```

Description: This is a subclass of model_ct_bundle, specialized for model datasets.

Parameters:

`a_dataset`: A params_cip_trace_fileset object.

`a_db`: The raw params_tests_db object created from the dataset. It only needs to have the pAcip, trial, and ItemIndex columns.

`a_ranked_db`: The one-model-per-line DB created from the raw DB.

`a_crit_bundle`: The bundle object associated with crit_db that caused the ranking in a_ranked_db.

`props`: A structure with any optional properties.

Returns a structure object with the following fields:

crit_bundle, model_ct_bundle.

See also: `model_ct_bundle` (p. 100), `ranked_db` (p. 176), `params_tests_dataset` (p. 113)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/12/13

A.16.2 Method `model_ranked_to_physiol_bundle/plotfICurve`

Usage:

```
a_doc = docfICurve(r_bundle, crit_bundle, crit_db, props)
```

Parameters:

`r_bundle`: A ranked_bundle object.

`rank_num`: Rank index for which to generate the a_doc.

`props`: A structure with any optional properties.

Returns:

a_doc: A doc_plot that contains a f-I curve plot and associated captions.

Example:

```
» a_d = docfICurve(r, 1);
» plot(a_d, 'The f-I curve of best matching model');
```

See also: `doc_generate` (p. 89), `doc_plot` (p. 93)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/01/16

A.16.3 Method `model_ranked_to_physiol_bundle/plotCompareRanks`

Summary: OBSOLETE - Generates a plots of given ranks from the ranked_bundle.

Usage:

```
plots = plotCompareRanks(r_bundle, crit_bundle, crit_db, props)
```

Parameters:

`r_bundle`: A ranked_bundle object.

`ranks`: Vector of rank indices for which to generate the plots.

`props`: A structure with any optional properties.

Returns:

plots: A structure that contains the joined_db, and the plot vectors trace_d100_plots and trace_h100_plots.

Example:

```
» plots = plotCompareRanks(r, 1:10);
» plotFigure(plots.trace_d100_plots(1), 'The best matching +100 pA CIP trace');
```

See also:

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/01/16

A.16.4 Method `model_ranked_to_physiol_bundle/comparisonReport`

Summary: OBSOLETE - Generates a report by comparing r_bundle with the given match criteria, crit_db from crit_bundle.

Usage:

```
a_doc_multi = comparisonReport(r_bundle, crit_bundle, crit_db, props)
```

Description: Generates a LaTeX document with: - (optional) Raw traces compared with some best matches at different distances - Values of some top matching a_db rows and match errors in a floating table. - colored-plot of measure errors for some top matches. - Parameter distributions of 50 best matches as a bar graph.

Parameters:

`r_bundle`**:** A dataset_db_bundle object that contains the DB to compare rows from.

`crit_bundle`**:** A dataset_db_bundle object that contains the criterion dataset.

`crit_db`**:** A tests_db object holding the match criterion tests and STDs which can be created with matchingRow.

`props`**:** A structure with any optional properties.

`caption`**:** Identification of the criterion db (not needed/used?).

`num_matches`**:** Number of best matches to display (default=10).

`rotate`**:** Rotation angle for best matches table (default=90).

Returns:

tex_string: LaTeX document string.

See also: `displayRowsTeX` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/01/17

A.17 Class `params_cip_trace_fileset`

A.17.1 Constructor `params_cip_trace_fileset/params_cip_trace_fileset`

Summary: Description of a raw dataset consisting of cip_trace files varying with parameter values.

Usage:

```
obj = params_cip_trace_fileset(file_pattern, dt, dy, pulse_time_start, pulse_time_width, id,
props)
```

Description: This is a subclass of params_tests_fileset.

Parameters:

`file_pattern`**:** File pattern mathing all files to be loaded.

`dt`**:** Time resolution [s]

`dy`**:** y-axis resolution [ISI (V, A, etc.)]

`pulse_time_start, pulse_time_width`**:** Start and width of the pulse [dt]

`id`**:** An identification string

`props`**:** A structure with any optional properties.

`profile_class_name`**:** Use this profile class (Default: 'cip_trace_profile'). (All other props are passed to cip_trace objects)

Returns a structure object with the following fields:

params_tests_fileset, pulse_time_start, pulse_time_width.

Example:

```
» fileset = params_cip_trace_fileset('/home/abc/data/*.bin', 1e-4, 1e-3, 20001, 10000, 'sim
dataset gpsc0501', struct('trace_time_start', 10001, 'type', 'sim', 'scale_y', 1e3))
```

See also: `params_tests_fileset` (p. 135), `params_tests_db` (p. 118)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.17.2 Method `params_cip_trace_fileset/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.17.3 Method `params_cip_trace_fileset/ctFromRows`

Summary: Loads a cip_trace object from raw data files in the fileset.

Usage:

```
a_cip_trace = ctFromRows(m_fileset, m_dball, a_db|itemIndices, cip_levels, props)
```

Parameters:

`m_fileset`**:** A physiol_cip_traceset_fileset object.

`m_dball`**:** A DB created by this fileset that contains the trial, pAcip, and ItemIndex cols.

`a_db`**:** A DB that has one trial for each cip_trace to be loaded.

`itemIndices`**:** A column vector with ItemIndex numbers.

`cip_levels`**:** A column vector of CIP-levels to be loaded.

`props`**:** A structure with any optional properties.

`neuronLabel`**:** appropriate unique neuron label generated by the bundle. (passed to params_cip_trace_fileset/cip_trace)

Returns:

a_cip_trace: One or more cip_trace objects that hold the raw data.

See also: `loadItemProfile` (p. **??**), `physiol_cip_traceset/cip_trace` (p. 150)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/07/13

A.17.4 Method `params_cip_trace_fileset/set`

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.17.5 Method `params_cip_trace_fileset/cip_trace_profile`

Summary: Loads a raw cip_trace_profile given a file_index to this fileset.

Usage:

```
a_cip_trace_profile = cip_trace_profile(fileset, file_index)
```

Parameters:

`fileset`**:** A params_tests_fileset.

`file_index`**:** Index of file in fileset.

Returns:

a_cip_trace_profile: A cip_trace_profile object.

See also: `cip_trace_profile` (p. 71), `params_tests_fileset` (p. 135)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.17.6 Method `params_cip_trace_fileset/display`

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.17.7 Method params_cip_trace_fileset/cip_trace

Summary: Loads raw cip_traces for each given file_index in this fileset.

Usage:

```
a_cip_trace = cip_trace(fileset, file_index|a_db, props)
```

Parameters:

- fileset: A params_tests_fileset.
- file_index: A single or array of indices of files in fileset.
- a_db: A DB created by this fileset to read the item indices from.
- props: A structure with any optional properties.
  - neuronLabel: Used for annotation purposes.

Returns:

a_cip_trace: A cip_trace object.

See also: cip_trace (p. 56), params_tests_fileset (p. 135)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/13

A.17.8 Method params_cip_trace_fileset/loadItemProfile

Summary: Loads a cip_trace_profile object from a raw data file in the fileset.

Usage:

```
[params_row, tests_row] = loadItemProfile(fileset, file_index)
```

Parameters:

- fileset: A params_tests_fileset.
- file_index: Index of file in fileset.

Returns:

a_profile: A profile object that implements the getResults method.

See also: itemResultsRow (p. ??), params_tests_fileset (p. 135), paramNames (p. ??), testNames (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.18 Class params_tests_dataset

A.18.1 Constructor params_tests_dataset/params_tests_dataset

Summary: Contains a set of data objects or files of raw data varying with parameter values.

Usage:

```
obj = params_tests_dataset(list, dt, dy, id, props)
```

Description: This is an abstract base class for keeping dataset information separate from the parameters-results database (params_tests_db). The list contents can be filenames or objects (such as cip_traces) from which to get the raw data. The dataset should have all the necessary information to create a db when needed. This is an abstract class, thet it it cannot act on its own. Only fully implemented subclasses can actually hold datasets. See methods below.

Parameters:

`list`**:** Array of dataset items (filenames, objects, etc.).

`dt`**:** Time resolution [s]

`dy`**:** y-axis resolution [integral V, A, etc.]

`id`**:** An identification string.

`props`**:** A structure with any optional properties.

`type`**:** type of file (default = ”)

Returns a structure object with the following fields:

list, dt, dy, id, props (see above).

See also: `params_tests_db` (p. 118), `params_tests_fileset` (p. 135), `cip_traces_dataset` (p. 72)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/12/02

A.18.2 Method params_tests_dataset/get

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.18.3 Method `params_tests_dataset/addItem`

Summary: Returns the new dataset with the added item.

Usage:

```
dataset = addItem(dataset, item)
```

Description: Note that, this is NOT the way to create a dataset. It is only intended for small additions to an existing dataset. This method is too slow for creating large datasets. The normal method for creating datasets is providing the full list of items to the class constructor.

Parameters:

`dataset`**:** A params_tests_dataset.

`item`**:** New item to add in dataset.

Returns:

dataset: With the added item.

See also: `itemResultsRow` (p. **??**), `params_tests_dataset` (p. 113), `paramNames` (p. **??**), `testNames` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/01/25

A.18.4 Method `params_tests_dataset/readDBItems`

Summary: Reads all items to generate a params_tests_db object.

Usage:

```
[params, param_names, tests, test_names] = readDBItems(obj, items)
```

Description: This is a generic method to convert from params_tests_fileset to a params_tests_db, or a subclass. This method depends on the paramNames, testNames, and itemResultsRow functions. Outputs of this function can be directly fed to the constructor of a params_tests_db or a subclass.

Parameters:

`obj`**:** A params_tests_fileset object.

`items`**:** (Optional) List of item indices to use to create the db.

Returns:

params, param_names, tests, test_names: See params_tests_db.

See also: `params_tests_db` (p. 118), `params_tests_fileset` (p. 135), `itemResultsRow` `testNames` (p. **??**), `paramNames` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/24

A.18.5 Method `params_tests_dataset/set`

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.18.6 Method `params_tests_dataset/getItemParams`

Summary: Get the parameter values of a dataset item.

Usage:

```
params_row = getItemParams(dataset, index, a_profile)
```

Description: This method can retrieve the item parameters by using either the dataset and the index to find the item or simply by using the item profile, a_profile.

Parameters:

`dataset`: A params_tests_dataset.

`index`: Index of item in dataset.

`a_profile`: An item profile.

Returns:

params_row: Parameter values in the same order of paramNames

See also: `itemResultsRow` (p. ??), `params_tests_dataset` (p. 113), `paramNames` (p. ??), `testNames` (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/10

A.18.7 Method `params_tests_dataset/testNames`

Summary: Returns the ordered names of tests for this dataset.

Usage:

```
test_names = testNames(dataset, item)
```

Description: Looks at the results of the first file to find the test names.

Parameters:

`dataset`: A params_tests_dataset.

Returns:

params_names: Cell array with ordered parameter names. item: (Optional) If given, read names by loading item at this index.

See also: params_tests_dataset (p. 113), paramNames (p. **??**), testNames (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/10

A.18.8 Method params_tests_dataset/subsref

Summary: Defines generic indexing for objects.

A.18.9 Method params_tests_dataset/subsasgn

Summary: Defines generic index-based assignment for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/02/06

A.18.10 Method params_tests_dataset/params_tests_db

Summary: Generates a params_tests_db object from the dataset.

Usage:

```
db_obj = params_tests_db(obj, items, props)
```

Description: This is a converter method to convert from params_tests_dataset to params_tests_db. Uses readDBItems to read the files. A customized subclass should provide the correct paramNames, testNames, and itemResultsRow functions. Adds a ItemIndex column to the DB to keep track of raw data files after shuffling.

Parameters:

obj: A params_tests_dataset object.

items: (Optional) List of item indices to use to create the db.

props: Any optional params to pass to params_tests_db.

Returns:

db_obj: A params_tests_db object.

See also: readDBItems (p. **??**), params_tests_db (p. 118), params_tests_dataset (p. 113), itemResultsRow testNames (p. **??**), paramNames (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/09

A.18.11 Method params_tests_dataset/display

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.18.12 Method `params_tests_dataset/itemResultsRow`

Summary: Processes a raw data file from the dataset and return its parameter and test values.

Usage:

```
[params_row, tests_row] = itemResultsRow(dataset, index)
```

Description: This method is designed to be reused from subclasses as long as the loadItemProfile method is properly overloaded. Adds an Index column to the DB to keep track of raw data items after shuffling.

Parameters:

`dataset`: A params_tests_dataset.

`index`: Index of file in dataset.

Returns:

params_row: Parameter values in the same order of paramNames tests_row: Test values in the same order with testNames

See also: `loadItemProfile` (p. ??), `params_tests_dataset` (p. 113), `paramNames` (p. ??), `testNames` (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/10

A.18.13 Method `params_tests_dataset/getItem`

Summary: Returns the dataset item at given index.

Usage:

```
item = getItem(dataset, index)
```

Parameters:

`dataset`: A params_tests_dataset.

`index`: Index of item in dataset.

Returns:

item: Object, filename, etc.

See also: `itemResultsRow` (p. ??), `params_tests_dataset` (p. 113), `paramNames` (p. ??), `testNames` (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/12/03

A.19 Class params_tests_db

A.19.1 Constructor params_tests_db/params_tests_db

Summary: A generic database of test results varying with parameter values, organized in a matrix format.

Description: This is a subclass of tests_db. Defines all operations on this structure so that subclasses can use them.

Parameters:

num_params: Number of parameters.

a_tests_db: A tests_db upon which to build the params_tests_db.

props: A structure with any optional properties.

Returns a structure object with the following fields:

tests_db num_params: Number of variable parameters in simulations.

See also: tests_db (p. 227), test_variable_db (N/I) (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/08

A.19.2 Method params_tests_db/getParamNames

Summary: Gets parameter column names.

Usage:

```
col_names = getParamNames(db)
```

Description: Convenience function that delegates to getColNames.

Parameters:

db: A params_tests_db object.

Returns:

col_names: A cell array of strings.

See also: tests_db/getColNames (p. 252), tests_db (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2008/04/03

A.19.3 Method `params_tests_db/matchingRow`

Summary: Creates a criterion database for matching the tests of a row.

Usage:

```
crit_db = matchingRow(a_db, row, props)
```

Description: Overloaded method for skipping parameter values. STD for param values will be NaNs.

Parameters:

`a_db`**:** A tests_db object.

`row`**:** A row index to match.

`props`**:** A structure with any optional properties.

`distDB`**:** Take the standard deviation from this db instead.

Returns:

crit_db: A tests_db with two rows for values and STDs.

See also: `tests_db/matchingRow` (p. 229), `rankMatching` (p. **??**), `tests_db` (p. 227), `tests2cols` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/06/13

A.19.4 Method `params_tests_db/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.19.5 Method `params_tests_db/joinRows`

Summary: Joins the rows of the given db with rows of with_db with matching RowIndex values.

Usage:

```
a_db = joinRows(db, with_db, props)
```

Description: Takes the desired columns in with_db with rows having a row index and joins them next to dedired columns from the current db. Assumes each row index only appears once in with_db. The created db preserves the ordering of with_db.

Parameters:

`db`**:** A param_tests_db object.

`with_db`**:** A tests_db object with a RowIndex column.

`props`**:** A structure with any optional properties.

`indexColName`**:** (Optional) Name of row index column (default='RowIndex').

Returns:

a_db: A params_tests_db object.

See also: `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/16

A.19.6 Method `params_tests_db/rankVsDB`

Summary: Generates a ranking DB by comparing rows of this db with the given test criteria.

Usage:

```
a_ranked_db = rankVsDB(a_db, crit_db)
```

Description: Distance is each measure difference divided by the STD in to_db, squared and summed. Returned DB contains only the selected tests from crit_db and the parameters from initial a_db.

Parameters:

`a_db`**:** A params_tests_db object to compare rows from.

`crit_db`**:** A tests_db object holding the match criterion tests and STDs which can be created with matchingRow.

Returns:

a_ranked_db: The created DB with original rows and a distance measure, in ascending order.

See also: `matchingRow` (p. **??**), `rankMatching` (p. **??**), `joinRows` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/20

A.19.7 Method `params_tests_db/plotParamsHists`

Summary: Create a horizontal plot_stack of parameter histograms.

Usage:

```
a_ps = plotParamsHists(a_db, title_str, props)
```

Description: Skips the 'ItemIndex' test.

Parameters:

- `a_db`: A params_tests_db object.
- `title_str`: (Optional) A string to be concatanated to the title.
- `props`: A structure with any optional properties.
  - `quiet`: Do not display the DB id on the plot title.
  - `barAxisProps`: passed to plotEqSpaced for each bar axis.

Returns:

a_ps: A horizontal plot_stack of plots

See also: `plot_stack` (p. 170), `paramsHists` (p. **??**), `plotEqSpaced` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/04/07

A.19.8 Method `params_tests_db/set`

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.19.9 Method `params_tests_db/scanParamAllRows`

Summary: Scans given parameter range for each row in DB.

Usage:

```
a_params_db = scanParamAllRows(a_db, param, min_val, max_val, num_levels, props)
```

Description: Produces rows by replacing the desired parameter value, in all rows of DB, with num_levels values between the given boundaries, min_val and max_val. This results in a DB with num_levels times more rows than the original DB. Then, makeGenesisParFile can be used to generate a parameter file from this DB to drive new simulations.

Parameters:

- `a_db`: A params_tests_db object whose first row is subject to modifications.
- `param`: The parameter to be varied (see tests2cols for param description).
- `min_val, max_val`: The low and high boundaries for the parameter value.
- `num_levels`: Number of levels to produce, including the boundaries.
- `props`: A structure with any optional properties.
  - `renameTrial`: If given, the 'trial' column is renamed to this name.

`levelFunc`**:** Use this function to get the parameter range with feval(levelFunc, min_val, max_val, num_levels). Example: 'logLevels'

Returns:

a_params_db: A db only with params.

Example:

```
Sets NaF to given range with 100 levels:
» naf_rows_db = scanParamAllRows(a_db(desired_rows, :), 'NaF', 0, 1000, 100);
```

See also: `makeGenesisParFile` (p. **??**), `scaleParamsOneRow` (p. **??**), `ranked_db/blockedDistances` (p. 182), `getParamRowIndices` (p. **??**), `logLevels` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/02/16

A.19.10 Method `params_tests_db/plotVarBoxMatrix`

Summary: Create a stack of parameter-test variation plots organized in a matrix.

Usage:

```
a_plot_stack = plotVarBoxMatrix(a_db, p_t3ds)
```

Description: Skips the 'ItemIndex' test.

Parameters:

`a_db`**:** A tests_db object.

`p_t3ds`**:** Cell array of invariant parameter databases.

Returns:

a_plot_stack: A plot_stack with the plots organized in matrix form

See also: `params_tests_profile` (p. 140), `plotVar` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/17

A.19.11 Method `params_tests_db/meanDuplicateParams`

Summary: Takes the mean of all measures for rows that have the same parameter columns.

Usage:

```
a_params_tests_db = meanDuplicateParams(db, props)
```

Description: Calls tests_db/meanDuplicateRows with params as main_cols and tests as rest_cols.

Parameters:

`db`: A params_tests_db object.

`props`: Structure with optional parameters.

Returns:

a_params_tests_db: The db object of with the means on page 1 and standard deviations on page 2.

See also: `tests_db/meanDuplicateRows` (p. 274), `tests_db/mean` (p. 248), `tests_db/std` (p. 246), `sortedUniqueValues` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/12/20

A.19.12 Method `params_tests_db/onlyRowsTests`

Summary: Returns a tests_db that only contains the desired tests and rows (and pages).

Usage:

```
obj = onlyRowsTests(obj, rows, tests, pages)
```

Description: Selects the given dimensions and returns in a new tests_db object. Makes sure num_params remains correct.

Parameters:

`obj`: A tests_db object.

`rows, tests`: A logical or index vector of rows, or cell array of names of rows. If ':', all rows. For names, regular expressions are supported if quoted with slashes (e.g., '/a.*/'). See tests2idx.

`pages`: (Optional) A logical or index vector of pages. ':' for all pages.

Returns:

obj: The new tests_db object.

See also: `subsref` (p. **??**), `tests_db` (p. 227), `test2idx` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/17

A.19.13 Method `params_tests_db/paramsCoefs`

Summary: Calculates a corrcoefs_db for each param and collects them in a cell array.

Usage:

```
p_coefs = paramsCoefs(a_db, p_t3ds)
```

Description: Skips the 'ItemIndex' test.

Parameters:

`a_db`**:** A tests_db object.

`p_t3ds`**:** Cell array of invariant parameter databases.

Returns:

p_coefs: A cell array of corrcoefs_dbs for each param in a_db.

See also: `params_tests_profile` (p. 140)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/17

A.19.14 Method `params_tests_db/unionCat`

Summary: Vertically concatenate two databases with different parameters or tests.

Description: The parameters and tests in the result are a union of both. Adds 0 for parameter and NaN for tests in the rows which didn't have the additional columns before.

Author: Li Su

A.19.15 Method `params_tests_db/subsref`

Summary: Defines generic indexing for objects.

A.19.16 Method `params_tests_db/displayRankingsTeX`

Summary: Generates and displays a ranking DB by comparing rows of a_db with the given match criteria.

Usage:

```
tex_string = displayRankingsTeX(a_db, crit_db, props)
```

Description: Generates a LaTeX document with: - Values of 10 best matching a_db rows in a floating table. - (optional) Raw traces compared with some best matches at different distances - Parameter distributions of 50 best matches as a bar graph.

Parameters:

`a_db`**:** A params_tests_db object to compare rows from.

`crit_db`**:** A tests_db object holding the match criterion tests and STDs which can be created with matchingRow.

`props`**:** A structure with any optional properties.

- `caption`**:** Identification of the criterion db (not needed/used?).
- `a_dataset`**:** Dataset for a_db.
- `a_dball`**:** The non-joined DB for for a_db.
- `crit_dataset`**:** Dataset for crit_db.
- `crit_dball`**:** Dataset for crit_db.
- `num_matches`**:** Number of best matches to display (default=10).
- `rotate`**:** Rotation angle for best matches table (default=90).

Returns:

tex_string: LaTeX document string.

See also: `rankVsDB` (p. **??**), `displayRowsTeX` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/20

A.19.17 Method `params_tests_db/display`

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.19.18 Method `params_tests_db/addParams`

Summary: Inserts new parameter columns to tests_db.

Usage:

```
obj = addParams(obj, param_names, param_columns)
```

Description: Adds new columns to the database and returns the new DB. This operation is expensive in the sense that the whole database matrix needs to be enlarged just to add a single new column. The method of allocating a matrix, filling it up, and then providing it to the tests_db constructor is the preferred method of creating tests_db objects. This method may be used for measures obtained by operating on raw measures.

Parameters:

`obj`**:** A tests_db object.

`param_names`**:** A cell array of param names to be added.

`param_columns`: Data matrix of columns to be added.

Returns:

obj: The tests_db object that includes the new columns.

See also: `allocateRows` (p. ??), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/10/11

A.19.19 Method `params_tests_db/makeGenesisParFile`

Summary: Creates a Genesis parameter file with all the parameter values in a_db.

Usage:

```
makeGenesisParFile(a_db, filename, props)
```

Description: For each a_db row, print the parameter names in a file formatted for Genesis.

Parameters:

`a_db`: A params_tests_db object.

`filename`: Genesis parameter file to be created.

`props`: A structure with any optional properties.

`trialStart`: If given, adds/replaces the trial parameter and counts forward.

`makeParamDesc`: If 1, put the parameter names in a parameter description file with
with a .txt extension.

Returns:

nothing.

Example:

```
» blocked_rows_db = makeModifiedParamDB(ranked_for_gps0501a_db, 1, [1, 2], 10, [-100 100]);
» makeGenesisParFile(blocked_rows_db, 'blocked_gps0501-03.par')
```

See also: `makeModifiedParamDB` (p. ??), `scanParamAllRows` (p. ??), `scaleParamsOneRow` (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/03/13

A.19.20 Method params_tests_db/reIndexNeurons

Summary: Re-index neurons with accending numbers.

Usage:

```
new_db = reIndexNeurons(a_db, startNum, colName)
```

Description: This is useful for avoiding NeuronId conflicts when concatenating two databases. It can also remove the unused number 'hole' (e.g. after deleting rows) and make the indices continuous.

Parameters:

a_db: a tests_db object

startNum: the starting index number (default = 1)

colName: the column name or number of neuron index. (default = 'NeuronId')

Returns:

new_db: a new database with new neuron index.

See also: physiol_bundle (p. 142), tests_db/physiol_bundle (p. 233)

Author: Li Su. 03/21/2008

A.19.21 Method params_tests_db/rankVsAllDB

Summary: Generates ranking DBs by comparing rows of a_db with each row of to_db.

Usage:

```
tex_string = rankVsAllDB(a_db, to_db, a_dataset, to_dataset)
```

Description: Distance is each measure difference divided by the STD in to_db, squared and summed. Returned DB contains only the selected to_tests and the parameters from initial DB.

Parameters:

a_db: A params_tests_db object to compare rows from.

to_db: A tests_db object to compare it with.

a_dataset: Dataset for a_db.

to_dataset: Dataset for crit_db.

Returns:

ranked_dbs: Array of created DBs with original rows and a distance measure, in ascending order. tex_string: A LaTeX string for all tables created.

See also: rankVsDB (p. ??), matchingRow (p. ??), rankMatching (p. ??), joinRows (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/12/10

A.19.22 Method params_tests_db/paramsParamsCoefs

Summary: Calculates a corrcoefs_db for each param from correlations of variant params and invariant param coefs and collects them in a cell array.

Usage:

```
pp_coefs = paramsParamsCoefs(a_db, p_t3ds, p_coefs)
```

Description: Skips the 'ItemIndex' test.

Parameters:

a_db: A tests_db object.

p_t3ds: Cell array of invariant parameter databases.

p_coefs: Cell array of tests coefficients for each parameter.

Returns:

pp_coefs: A cell array of corrcoefs_dbs for each param combination in a_db.

See also: params_tests_profile (p. 140)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/17

A.19.23 Method params_tests_db/mergeMultipleCIPsInOne

Summary: Merges multiple rows with different CIP data into one, generating a database of one row per neuron.

Usage:

```
a_db = mergeMultipleCIPsInOne(db, names_tests_cell, index_col_name, props)
```

Description: It calls invarParam to separate DB into pages with different CIP level data. Then uses the names_tests_cell to choose tests from each page to be merged into the final database row. The tests will be suffixed with the field name so that they can be distinguished. RowIndex columns will be automatically included, and one of them can be chosen with index_col_name that has values for all cells. The suffixed for needs to be used to choose index_col_name, such as 'RowIndex_H100pA', assuming 'H100pA' was the field name in names_tests_cell that corresponds to page -100 pA.

Parameters:

`db`**:** A params_tests_db object.

`names_tests_cell`**:** A cell array alternating suffix names and test column vectors.

The order of names correspond to each unique CIP level in db, with increasing order.

`index_col_name`**:** (Optional) Name of row index column

(default is 'RowIndex' suffixed with the first field name).

`props`**:** A structure with any optional properties.

`cipLevels`**:** In case db is missing some levels, provides a list of cip levels that correspond to names_tests_cell db. Missing levels are replaced with NaN values. DB is filtered to remove other CIP levels.

Returns:

a_db: A params_tests_db object of organized values.

Example:

```
» control_phys_sdb =
mergeMultipleCIPsInOne(control_phys_db,
struct('_H100pA', [1:10], '_D100pA', [1:10 16:18]),
'RowIndex_H100pA')
```

See also: `invarValues` (p. **??**), `tests_3D_db` (p. 220), `corrCoefs` (p. **??**), `tests_3D_db/plotVarBox` (p. 224)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/01/13

A.19.24 Method `params_tests_db/getParamRowIndices`

Summary: Returns indices of rows with matching parameter values from rows of this db.

Usage:

```
row_indices = getParamRowIndices(a_db, rows, to_db)
```

Parameters:

`a_db`**:** A params_tests_db object.

`rows`**:** rows to find indices for.

`to_db`**:** Where to find the matching rows.

Returns:

row_indices: Array of row indices.

See also: makeModifiedParamDB (p. ??), scanParamAllRows (p. ??), scaleParamsOneRow (p. ??), makeGenesisParFile (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/01/14

A.19.25 Method params_tests_db/testsHists

Summary: Calculates histograms for all tests and returns them in a cell array.

Usage:

```
t_hists = testsHists(a_db, num_bins)
```

Description: Skips the 'ItemIndex' test.

Parameters:

a_db**:** One or more tests_db objects in an array.

num_bins**:** Number of histogram bins (Optional, default=100), or vector of histogram bin centers.

Returns:

t_hists: An array of histograms for each test in a_db.

See also: params_tests_profile (p. 140)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/17

A.19.26 Method params_tests_db/getDualCIPdb

Summary: Generates a database by merging selected tests of depolarizing and hyperpolarizing cip results.

Usage:

```
a_db = getDualCIPdb(db, depol_tests, hyper_tests, depol_suffix, hyper_suffix)
```

Description: depol_tests need to have the RowIndex column in it.

Parameters:

db**:** A params_tests_db object.

Returns:

a_db: A params_tests_db object of organized values.

Example:

```
» control_phys_sdb = getDualCIPdb(control_phys_db, depol_tests, hyper_tests, ”, 'Hyp100pA')
where depol_tests and hyper_tests are cell arrays of selected tests.
```

See also: invarValues (p. ??), tests_3D_db (p. 220), corrCoefs (p. ??), tests_3D_db/plotPair (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/01/13

A.19.27 Method params_tests_db/invarParam

Summary: Generates a 3D database of invariant values of a parameter and all test columns.

Usage:

```
a_3D_db = invarParam(db, param)
```

Description: Finds all combinations when the rest of the parameters are fixed, and saves the variation of the selected parameter and all tests in a new database.

Parameters:

db: A tests_db object.

param: A parameter name/column number

Returns:

a_3D_db: A tests_3D_db object of organized values.

See also: invarValues (p. ??), tests_3D_db (p. 220), corrCoefs (p. ??), tests_3D_db/plotPair (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/07

A.19.28 Method params_tests_db/getProfile

Summary: Create a profile object from a params_tests_db by collecting statistics.

Usage:

```
a_pt_profile = getProfile(a_db, props)
```

Description: Calculates the following results items: idx: Name-index pairs for accessing results arrays. t_hists: Cell array of histograms of each test. p_t3ds: Cell array of invariant relations of each parameter with all tests. pt_hists: Cell array of separate test value histograms for uniques value of each parameter. p_stats: Cell array of test stats for each param. p_coefs: Cell array of correlation coefficients for each parameter with all tests. pt_coefs_hists: Cell matrix of histograms of coefficients from correlations of each parameter with each test. pp_coefs: Cell 3D matrix of mean coefficients from correlations of each parameter with correlation coefficients of each parameter with each test.

Parameters:

`a_db`: A params_tests_db object.

`props`: A structure with any optional properties.

Returns a params_tests_profile object.

See also: `params_tests_profile` (p. 140), `results_profile` (p. 182), `params_tests_db` (p. 118), `params_tests_fileset` (p. 135), `tests_db` (p. 227), `tests_3D_db` (p. 220), `histogram_db` (p. 96), `stats_db` (p. 214), `corrcoefs_db` (p. 82)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/13

A.19.29 Method `params_tests_db/delColumns`

Summary: Deletes columns from tests_db.

Usage:

```
index = delColumns(obj, tests)
```

Description: Overloaded function that maintains correct number of parameters. See original tests_db/delColumns.

Parameters:

`obj`: A tests_db object.

`tests`: Numbers or names of tests (see tests2cols)

Returns:

obj: The tests_db object that is missing the columns.

See also: `tests_db/delColumns` (p. 274)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/10/11

A.19.30 Method `params_tests_db/scaleParamsOneRow`

Summary: Scales chosen parameters in a row by multiplying with levels to create a new parameter db with as many rows as values in levels.

Usage:

```
a_params_db = scaleParamsOneRow(a_db, params, levels)
```

Description: Produces rows by multiplying desired params, in the first row of DB, with each value in levels. Then, makeGenesisParFile can be used to generate a parameter file from this DB to drive new simulations.

Parameters:

`a_db`**:** A params_tests_db object whose first row is subject to modifications.

`params`**:** Parameters to be varied (see tests2cols for param description).

`levels`**:** Column vector of parameter value multipliers (1=unity).

Returns:

a_params_db: A db only with params.

Example:

```
Blocks NaF from 0» naf_rows_db = scanOneParam(a_db(desired_row, :), 'NaF', 0:0.1:1);
```

See also: `ranked_db/blockedDistances` (p. 182), `getParamRowIndices` (p. **??**), `makeGenesisParFile` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/02/16

A.19.31 Method `params_tests_db/paramsHists`

Summary: Calculates histograms for all parameters and returns in a cell array.

Usage:

```
p_hists = paramsHists(a_db)
```

Description: Skips the 'ItemIndex' test. Useful for looking at subset databases and find out what parameter values are used most.

Parameters:

`a_db`**:** A tests_db object.

Returns:

p_hists: An array of histograms for each parameter in a_db.

See also: `params_tests_profile` (p. 140)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/20

A.19.32 Method `params_tests_db/paramsTestsCoefsHists`

Summary: Calculates histograms for all pairs of params and tests coefficients and returns in a cell array.

Usage:

```
pt_coefs_hists = paramsTestsCoefsHists(a_db, p_coefs)
```

Description: Skips the 'ItemIndex' test.

Parameters:

`a_db`: A tests_db object.

`p_coefs`: Cell array of tests coefficients for each parameter.

Returns:

pt_coefs_hists: A cell array of corrcoefs_dbs for each param in a_db.

See also: `params_tests_profile` (p. 140)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/17

A.19.33 Method `params_tests_db/crossProd`

Summary: Create a DB by taking the cross product of two database row sets.

Usage:

```
cross_db = crossProd(a_db, b_db)
```

Description: Overloaded function to maintain correct number of parameters after cross product operation. See original in tests_db/crossProd.

Parameters:

`a_db, b_db`: A tests_db object.

Returns:

cross_db: The tests_db object with all combinations of rows.

See also: `tests_db/crossProd` (p. 276)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/10/11

A.19.34 Method `params_tests_db/invarParams`

Summary: Calculates invariant param dbs for all parameters and returns in an array.

Usage:

```
p_t3ds = invarParams(a_db)
```

Description: Skips the 'ItemIndex' test.

Parameters:

`a_db`**:** A tests_db object.

Returns:

p_t3ds: An array of tests_3D_dbs for each param in a_db.

See also: `params_tests_profile` (p. 140)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/17

A.20 Class `params_tests_fileset`

A.20.1 Constructor `params_tests_fileset/params_tests_fileset`

Summary: Description of a set of data files of raw data varying with parameter values.

Usage:

```
obj = params_tests_fileset(file_pattern, dt, dy, id, props)
```

Description: This is a subclass of params_tests_dataset. This class is used to generate params_tests_db objects and keep a connection to the raw data files. This class only keeps names of files and loads raw data files whenever it's requested. A database object can easily be generated using the convertion methods. Most methods defined here can be used as-is, however some should be overloaded in subclasses. The specific methods are loadItemProfile.

Parameters:

`file_pattern`**:** File pattern, or cell array of patterns, matching all files to be loaded.

`dt`**:** Time resolution [s]

`dy`**:** y-axis resolution [ISI (V, A, etc.)]

`id`**:** An identification string

`props`**:** A structure with any optional properties.

`num_params`**:** Number of parameters that appear in filenames.

param_trial_name: Use this name on the filename as the 'trial' parameter.

param_row_filename: If given, the 'trial' parameter will be used to address rows from this file and acquire parameters.

param_desc_filename: Contains the parameter range descriptions one per
each row. The parameter names are acquired from this file.

param_names: Cell array of parameter names corresponding to the param_row_filename columns can be specified as an alternative to specifying param_desc_filename. These names are not for the parameters present in the data filename.

profile_method_name: It can be one of the profile-creating methods in this
class. E.g., 'trace_profile', 'srp_trace_profile', etc. (See parent classes and cip_trace object for more props)

Returns a structure object with the following fields:

params_tests_dataset, path: The pathname to files.

See also: `params_tests_db` (p. 118), `tests_db` (p. 227), `test_variable_db (N/I)` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/09

A.20.2 Method `params_tests_fileset/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.20.3 Method `params_tests_fileset/set`

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.20.4 Method `params_tests_fileset/trace_profile`

Summary: Loads a raw trace_profile given a file_index to this fileset.

Usage:

```
a_trace_profile = trace_profile(fileset, file_index)
```

Parameters:

`fileset`: A params_tests_fileset.

`file_index`: Index of file in fileset.

Returns:

a_trace_profile: A trace_profile object.

See also: `trace_profile` (p. 288), `params_tests_fileset` (p. 135)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/13

A.20.5 Method `params_tests_fileset/getItemParams`

Summary: Get the parameter values of a dataset item.

Usage:

```
params_row = getItemParams(dataset, index)
```

Parameters:

`dataset`: A params_tests_dataset.

`index`: Index of item in dataset.

Returns:

params_row: Parameter values in the same order of paramNames

See also: `itemResultsRow` (p. **??**), `params_tests_dataset` (p. 113), `paramNames` (p. **??**), `testNames` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/12/03

A.20.6 Method `params_tests_fileset/addFiles`

Summary: Adds to existing list of files in set.

Usage:

```
[a_fileset, index_list] = addFiles(a_fileset, file_pattern, props)
```

Parameters:

`a_fileset`: A params_tests_fileset object.

`file_pattern`: File pattern, or cell array of patterns, matching additional files.

`props`: A structure with any optional properties.

`param_row_filename`**:** Update parameters from here. The 'trial' parameter is used
to address rows from this file and acquire parameters.

Returns:

a_fileset: The augmented fileset object. index_list: The vector of index numbers of the new files added. Can be used to selectively load the new files into a DB using params_test_db.

See also: `params_tests_fileset` (p. 135), `params_tests_dataset/params_test_db.` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/02/01

A.20.7 Method `params_tests_fileset/display`

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.20.8 Method `params_tests_fileset/loadItemProfile`

Summary: Loads a profile object from a raw data file in the fileset.

Usage:

```
a_profile = loadItemProfile(fileset, file_index)
```

Description: Subclasses should overload this function to load the specific profile object they desire. The profile class should define a getResults method which is used in the itemResultsRow method.

Parameters:

`fileset`**:** A params_tests_fileset.

`file_index`**:** Index of file in fileset.

Returns:

a_profile: A profile object that implements the getResults method.

See also: `itemResultsRow` (p. **??**), `params_tests_fileset` (p. 135), `paramNames` (p. **??**), `testNames` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.20.9 Method params_tests_fileset/paramNames

Summary: Returns the ordered names of parameters for this fileset.

Usage:

```
param_names = paramNames(fileset, item)
```

Description: Looks at the filename of the first file to find the parameter names.

Parameters:

fileset: A params_tests_fileset.

item: (Optional) If given, read param names by loading item at this index.

Returns:

params_names: Cell array with ordered parameter names.

See also: params_tests_fileset (p. 135), paramNames (p. **??**), testNames (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/10

A.20.10 Method params_tests_fileset/trace

Summary: Loads a raw trace given a file_index to this fileset.

Usage:

```
a_trace = trace(fileset, file_index)
```

Parameters:

fileset: A params_tests_fileset.

file_index: Index of file in fileset.

Returns:

a_trace: A trace object.

See also: trace (p. 278), params_tests_fileset (p. 135)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/13

A.21 Class params_tests_profile

A.21.1 Constructor params_tests_profile/params_tests_profile

Summary: Holds the results profile from a params_tests_db.

Usage:

```
a_pt_profile = params_tests_profile(results, a_db, props)
```

Parameters:

a_db**:** A params_tests_db object.

results**:** A structure containing test results.

props**:** A structure with any optional properties.

Returns a structure object with the following fields:

results_profile: Contains results of tests. db: The params_tests_db. props.

See also: results_profile (p. 182), params_tests_db/params_tests_profile (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/13

A.21.2 Method params_tests_profile/get

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.22 Class period

A.22.1 Constructor period/period

Summary: Start and end times of a period in terms of the dt of the trace to which belongs.

Usage:

```
obj = period(start_time, end_time)
```

Parameters:

(see below for the rest)

Returns a structure object with the following fields:

start_time, end_time: Inclusive period [dt].

See also: trace (p. 278), spikes (p. 202), spike_shape (p. 189)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/07/30

A.22.2 Method `period/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.22.3 Method `period/SpikeTimesinPeriod`

Usage:

```
SpkTimes=Interval(times, period)
```

Parameters:

`times`: an array of spike times.

`period`: A period object

Returns:

the_period: The cropped set of spike times that fall within a period.

See also: `period` (p. 140), `cip_trace` (p. 56), `trace` (p. 278), `spikes` (p. 202)

Author: Tom Sangrey, 2006/01/26

A.22.4 Method `period/set`

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.22.5 Method `period/subsref`

Summary: Defines generic indexing for objects.

A.22.6 Method `period/display`

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.23 Class physiol_bundle

A.23.1 Constructor physiol_bundle/physiol_bundle

Summary: The physiology dataset and the DB created from it bundled together.

Usage:

```
a_bundle = physiol_bundle(a_cell, props)
```

Description: This is a subclass of dataset_db_bundle, specialized for physiology datasets.

Parameters:

a_cell: A cell array that contains the following elements:

a_dataset: A cell-enclosed physiol_cip_traceset_fileset object.

a_db: The raw params_tests_db object created from the dataset.
It only needs to have the pAcip, pAbias, TracesetIndex, and ItemIndex columns.

a_joined_db: The one-treatment-per-line DB created from the raw DB.

props: A structure with any optional properties.

controlDB: Use this as the ontrol DB rather than computing.

Returns a structure object with the following fields:

dataset_db_bundle, joined_control_db: DB of control neurons (no pharmacological applications).

See also: dataset_db_bundle (p. 83), tests_db (p. 227), params_tests_dataset (p. 113)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/12/13

A.23.2 Method physiol_bundle/matchingControlNeuron

Summary: Creates a criterion database for matching the neuron at traceset_index.

Usage:

```
a_crit_bundle = matchingControlNeuron(a_bundle, neuron_id, props)
```

Description: Copies selected test values from row as the first row into the criterion db. Adds a second row for the STD of each column in the db.

Parameters:

a_bundle: A physiol_bundle object.

neuron_id: A NeuronId of the neuron to match.

props: A structure with any optional properties.

Returns:

a_crit_bundle: A tests_db with two rows for values and STDs.

Example:

```
Matches gpd0421c from cip_traces_all_axoclamp.txt:
» a_crit_bundle = matchingControlNeuron(pbundle, 33)
(see example in matchingRow)
```

See also: `rankMatching` (p. **??**), `tests_db` (p. 227), `tests2cols` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/12/21

A.23.3 Method `physiol_bundle/matchingRow`

Summary: Creates a criterion database for matching the neuron at traceset_index.

Usage:

```
a_crit_db = matchingRow(p_bundle, traceset_index, props)
```

Description: Copies selected test values from row as the first row into the criterion db. Adds a second row for the STD of each column in the db.

Parameters:

`p_bundle`**:** A physiol_bundle object.

`traceset_index`**:** A TracesetIndex of the neuron and treatments to match.

`props`**:** A structure with any optional properties.

Returns:

a_crit_db: A tests_db with two rows for values and STDs.

Example:

```
physiol_bundle has an overloaded matchingRow method that
takes the TracesetIndex as argument:
» a_crit_bundle = matchingRow(pbundle, 61)
» a_ranked_bundle = rankMatching(mbundle, a_crit_bundle);
» printTeXFile(comparisonReport(a_ranked_bundle), 'my_report.tex')
```

See also: `rankMatching` (p. **??**), `tests_db/matchingRow` (p. 229)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/12/21

A.23.4 Method physiol_bundle/get

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.23.5 Method physiol_bundle/getNeuronRowIndex

Summary: Returns the neuron index from bundle.

Usage:

```
a_row_index = getNeuronRowIndex(a_bundle, traceset_index, props)
```

Parameters:

a_bundle: A physiol_bundle object.

traceset_index: The TracesetIndex number of neuron, or a DB row containing this.

props: A structure with any optional properties.

Returns:

a_row_index: A row index of neuron in a_bundle.joined_db.

See also: dataset_db_bundle/getNeuronRowIndex (p. 84)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/06/09

A.23.6 Method physiol_bundle/ctFromRows

Summary: Loads a cip_trace object from a raw data file in the a_pbundle.

Usage:

```
a_cip_trace = ctFromRows(a_pbundle, a_db|traceset_idx, cip_levels, props)
```

Parameters:

a_pbundle: A physiol_cip_traceset_fileset object.

a_db: A DB created by this fileset to read the traceset indices from.

traceset_idx: A column vector with traceset indices.

cip_levels: A column vector of CIP-levels to be loaded.

props: A structure with any optional properties.

traces: column vector of trace indices to load.

showParamsList: Cell array of params or treatments to include in the id field.

Returns:

a_cip_trace: One or more cip_trace object that holds the raw data.

See also: `loadItemProfile` (p. **??**), `physiol_cip_traceset/cip_trace` (p. 150)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/07/13

A.23.7 Method `physiol_bundle/getNeuronLabel`

Summary: Constructs the neuron label from dataset.

Usage:

```
a_label = getNeuronLabel(a_bundle, traceset_index, props)
```

Parameters:

`a_bundle`**:** A physiol_cip_traceset_fileset object.

`traceset_index`**:** The traceset index of neuron.

`props`**:** A structure with any optional properties.

Returns:

a_label: A string label identifying selected neuron in bundle.

See also: `dataset_db_bundle` (p. 83)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/05/05

A.23.8 Method `physiol_bundle/plotfICurveStats`

Summary: Generates a f-I curve mean-std plot of physiology DB.

Usage:

```
a_plot = plotfICurveStats(p_bundle, title_str, props)
```

Parameters:

`p_bundle`**:** A physiol_bundle object.

`title_str`**:** (Optional) String to append to plot title.

`props`**:** A structure with any optional properties.

`quiet`**:** if given, no title is produced
(passed to plot_superpose)

Returns:

a_plot: An f-I curve plot.

Example:

```
» plotFigure(plotfICurveStats(pbundle));
```

See also: `dataset_db_bundle/plotfICurve` (p. 88), `plot_abstract` (p. 157), `plot_superpose` (p. 173)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/06/16

A.23.9 Method `physiol_bundle/set`

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.23.10 Method `physiol_bundle/constrainedMeasuresPreset`

Summary: Returns a physiol_bundle with constrained measures according to chosen preset.

Usage:

```
[a_pbundle test_names] = constrainedMeasuresPreset(a_pbundle, preset, props)
```

Parameters:

`a_pbundle`: A physiol_cip_traceset_fileset object.

`preset`: Choose preset measure list (default=1).

`props`: A structure with any optional properties.

Returns:

a_pbundle: One or more cip_trace object that holds the raw data.

See also: `loadItemProfile` (p. **??**), `physiol_cip_traceset/cip_trace` (p. 150)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/01/19

A.23.11 Method `physiol_bundle/bestMatchAllNeurons`

Summary: Finds the best match among given database for each physiology neuron.

Usage:

```
all_ranks_db = bestMatchAllNeurons(p_bundle, joined_db, props)
```

Description: Returns a database of best matching entries from joined_db for each entry in p_bundle.joined_control_db.

Parameters:

`p_bundle`: A physiol_bundle object.

`joined_db`: A database with neuron representations to rank against neurons.

`props`: A structure with any optional properties.
(passed to rankMatching)

Returns:

all_ranks_db: DB of best matching from joined_db. Each row corresponds to p_bundle.joined_control_db rows.

Example:

```
» all_ranks_db = ...
bestMatchAllNeurons(constrainedMeasuresPreset(pbundle2, 6), mbundle_maxcond.joined_db)
» plotXRows(all_ranks_db, 'Distance', 'maxcond DB distance per neuron', 'maxcond', ...
struct('LineStyle', '-', 'quiet', 1, 'PaperPosition', [0 0 4 3]))
```

See also: `tests_db/rankMatching` (p. 249), `tests_db/matchingRow` (p. 229)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/05/24

A.24 Class `physiol_cip_traceset`

A.24.1 Constructor `physiol_cip_traceset/physiol_cip_traceset`

Summary: Dataset of cip traces from same PCDX file.

Usage:

```
obj = physiol_cip_traceset(trace_str, data_src, chaninfo, dt, dy, treatments, neuron_id,
props);
```

Description: This is a subclass of params_tests_dataset. Each trace varies in bias, pulse times and cip magnitude.

Parameters:

`trace_str`: Trace list in the format for loadtraces or just a Matlab vector.

`data_src`: Absolute path of PCDX data source.

`chaninfo`: 4-element array containing vchan, ichan, vgain, igain

`vchan, ichan`: Current and voltage channels.

`vgain, igain`: External gain factors for voltage channel and current channel (vgain does NOT include the 10X amplification from the Axoclamp, so vgain = 1 would mean no additional amplification beyond the 10X.)

dt: Time resolution [s].

dy: Y-axis resolution [V] or [A].

treatments: Structure containing the names and concentrations of compounds.

neuron_id: Neuron name.

props: A structure with any optional properties.

- **nsHDF5:** For NeuroSAGE HDF5 files, processing is faster if the output of ns_open_file is given here. Must be defined to allow special NeuroSAGE processing.
- **profile_class_name:** Use this cip_trace function to return a profile (Default: 'getProfileAllSpikes').
- **cip_list:** Vector of cip levels to which the current trace will be matched. (All other props are passed to cip_trace objects)

Returns a structure object with the following fields:

params_tests_dataset, data_src, ichan, vchan, vgain, igain, treatments.

See also: `cip_traces` (p. **??**), `params_tests_dataset` (p. 113), `params_tests_db` (p. 118)

Author: Cengiz Gunay <cgunay@emory.edu> and Thomas Sangrey, 2005/01/17

A.24.2 Method `physiol_cip_traceset/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.24.3 Method `physiol_cip_traceset/setProp`

Summary: Generic method for setting optional object properties.

Usage:

```
obj = setProp(obj, prop1, val1, prop2, val2, ...)
```

Description: Modifies or adds property values. As many property name-value pairs can be specified.

Parameters:

obj: Any object that has a props field.

attr: Property name

val: Property value.

Returns:

obj: The new object with the updated properties.

See also:

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/22

A.24.4 Method `physiol_cip_traceset/CIPform`

Summary: Extracts current bias and pulse information from the current channel.

Usage:

```
[ciptype, on, off, finish, bias, pulse] = ns_CIPform(traceset,trace_index)
```

Parameters:

`traceset`: A physiol_cip_traceset object.

`trace_index`: Index of item in traceset

See also: `cip_traces` (p. **??**), `params_tests_dataset` (p. 113), `params_tests_db` (p. 118)

Author: Thomas Sangrey, 2005

A.24.5 Method `physiol_cip_traceset/set`

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.24.6 Method `physiol_cip_traceset/getItemParams`

Summary: Get the parameter values of a dataset item.

Usage:

```
params_row = getItemParams(dataset, index, a_profile)
```

Parameters:

`dataset`: A params_tests_dataset.

`index`: Index of item in dataset.

`a_profile`: cip_trace_profile object

Returns:

params_row: Parameter values in the same order of paramNames

See also: itemResultsRow (p. **??**), params_tests_dataset (p. 113), paramNames (p. **??**), testNames (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/12/06

A.24.7 Method physiol_cip_traceset/cip_trace_profile

Summary: Loads a cip_trace_profile object from a raw data file in the traceset.

Usage:

```
a_profile = cip_trace_profile(traceset, trace_index)
```

Parameters:

`traceset`**:** A physiol_cip_traceset object.

`trace_index`**:** Index of file in traceset.

Returns:

a_profile: A profile object that implements the getResults method.

See also: itemResultsRow (p. **??**), params_tests_fileset (p. 135), paramNames (p. **??**), testNames (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu> and Thomas Sangrey, 2005/01/18

A.24.8 Method physiol_cip_traceset/cip_trace

Summary: Loads a cip_trace object from a raw data file in the traceset.

Usage:

```
a_cip_trace = cip_trace(traceset, trace_index, props)
```

Parameters:

`traceset`**:** A physiol_cip_traceset object.

`trace_index`**:** Index of file in traceset.

`props`**:** A structure with any optional properties.

- `showParamsList`**:** Cell array of params to add to id field.
- `showName`**:** Show the name of the cell in the id field (default=1).
- `TracesetIndex`**:** Indicates in the id field.

Returns:

a_cip_trace: A cip_trace object that holds the raw data.

See also: `itemResultsRow` (p. **??**), `params_tests_fileset` (p. 135), `paramNames` (p. **??**), `testNames` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/07/13

A.24.9 Method `physiol_cip_traceset/loadItemProfile`

Summary: Loads a cip_trace_profile object from a raw data file in the traceset.

Usage:

```
a_profile = loadItemProfile(traceset, trace_index)
```

Parameters:

`traceset`: A physiol_cip_traceset object.

`trace_index`: Index of file in traceset.

Returns:

a_profile: A profile object that implements the getResults method.

See also: `itemResultsRow` (p. **??**), `params_tests_fileset` (p. 135), `paramNames` (p. **??**), `testNames` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.24.10 Method `physiol_cip_traceset/itemResultsRow`

Summary: Processes a raw data file from the dataset and return its parameter and test values.

Usage:

```
[params_row, tests_row] = itemResultsRow(dataset, index)
```

Description: This method is designed to be reused from subclasses as long as the loadItemProfile method is properly overloaded. Adds an Index column to the DB to keep track of raw data items after shuffling.

Parameters:

`dataset`: A params_tests_dataset.

`index`: Index of file in dataset.

Returns:

params_row: Parameter values in the same order of paramNames tests_row: Test values in the same order with testNames

See also: `loadItemProfile` (p. ??), `params_tests_dataset` (p. 113), `paramNames` (p. ??), `testNames` (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/10

A.24.11 Method `physiol_cip_traceset/paramNames`

Summary: Returns the parameter names for this traceset.

Usage:

```
param_names = paramNames(traceset)
```

Description: Looks at the filename of the first file to find the parameter names.

Parameters:

`traceset`: A params_tests_dataset.

Returns:

param_names: Cell array with ordered parameter names.

See also: `params_tests_dataset` (p. 113), `paramNames` (p. ??), `testNames` (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/12/06

A.25 Class `physiol_cip_traceset_fileset`

A.25.1 Constructor `physiol_cip_traceset_fileset/physiol_cip_traceset_fileset`

Summary: Physiological fileset of traceset objects (concatenated).

Usage:

```
obj = physiol_cip_traceset_fileset(traceset_items, dt, dy, props)
```

Description: This is a subclass of params_tests_dataset. It contains a set of physiol_cip_traceset items that are tied to physical data sources. Each traceset can load a set of traces for an experimental recording. Most flexible usage is obtained when the input traceset_items is given as a cell array of physiol_cip_traceset objects. These objects can each link to PCDX or NeuroSAGE HDF5 files independent of each other. A regular Matlab script can be used to create such a cell array. If a function is defined to return such an array, it can be passed as traceset_items. Alternatively, the cell array can be constructed from an ASCII file as described below, such as for deprecated PCDX data files.

Parameters:

`traceset_items`**:** It can be a function handle, cell array or filename string. Function should return a cell array of physiol_cip_traceset items. Alternatively a preconstructed cell array can be provided directly. If it is an ASCII filename, then it should contain the following tab-delimited items: 1. Neuron ID (name to associate with the neuron). If left blank, use the filename with the '.all' extension removed. 2. The absolute path of the data file 3. The trace numbers to load, space-delimited (e.g. 1-21 24 26 27) 4. Vchan: voltage channel number 5. Ichan: current channel number 6. Vgain: external gain on voltage channel IN ADDITION to the 10X that automatically comes from the Axoclamp 2B. 7. Igain: external gain on current channel. 8. Pairs of condition names and molar concentrations in any order e.g.: TTX 1e-8 apamin 2e-7 picrotoxin 1e-4

Returns a structure object with the following fields:

neuron_idx: A structure that points from neuron names to NeuronId numbers. params_tests_dataset

See also: `physiol_cip_traceset` (p. 147), `params_tests_dataset` (p. 113), `params_tests_db` (p. 118)

Author: Cengiz Gunay <cgunay@emory.edu> and Thomas Sangrey, 2005/01/17

A.25.2 Method `physiol_cip_traceset_fileset/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.25.3 Method `physiol_cip_traceset_fileset/setProp`

Summary: Generic method for setting optional object properties.

Usage:

```
obj = setProp(obj, prop1, val1, prop2, val2, ...)
```

Description: Modifies or adds property values. As many property name-value pairs can be specified.

Parameters:

`obj`**:** Any object that has a props field.

`attr`**:** Property name

`val`**:** Property value.

Returns:

obj: The new object with the updated properties.

See also:

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/22

A.25.4 Method `physiol_cip_traceset_fileset/physiol_bundle`

Summary: Loads the database and then creates the physiol_bundle object.

Usage:

```
a_pbundle = physiol_bundle(fileset, props)
```

Description: Calls params_tests_db to get the db, and then calls tests_db/physiol_bundle to do transformations.

Parameters:

`fileset`: A physiol_cip_traceset_fileset object.

`props`: A structure with any optional properties.
(Passed to tests_db/physiol_bundle)

Returns:

a_physiol_bundle: One or more physiol_bundle object that holds the raw data.

See also: `tests_db/physiol_bundle` (p. 233)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/12/21

A.25.5 Method `physiol_cip_traceset_fileset/readDBItems`

Summary: Reads all items to generate a params_tests_db object.

Usage:

```
[params, param_names, tests, test_names] = readDBItems(obj, items)
```

Description: This is a specific method to convert from physiol_cip_traceset_fileset to a params_tests_db, or a subclass. Outputs of this function can be directly fed to the constructor of a params_tests_db or a subclass.

Parameters:

`obj`: A physiol_cip_traceset_fileset

`items`: (Optional) List of item indices to use to create the db.

Returns:

params, param_names, tests, test_names: See params_tests_db.

See also: params_tests_db (p. 118), params_tests_fileset (p. 135), itemResultsRow (p. ??)

A.25.6 Method physiol_cip_traceset_fileset/set

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.25.7 Method physiol_cip_traceset_fileset/display

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.25.8 Method physiol_cip_traceset_fileset/vertcat

Summary: Concatenates multiple physiol_cip_traceset_fileset objects.

Usage:

```
obj = vertcat(obj, obj2)
```

Description: Concatenates the list contents, and combines the neuron_idx structures. The properties such as dt, dy and props are retained from first object.

Parameters:

`obj, obj2`**:** Two physiol_cip_traceset_fileset objects without overlapping neuron_id items.

Returns:

obj: The new object with combined contents.

See also: physiol_cip_traceset_fileset (p. 152)

Author: Cengiz Gunay <cgunay@emory.edu>, 2008/01/13

A.25.9 Method `physiol_cip_traceset_fileset/cip_trace`

Summary: Loads a cip_trace object from a raw data file in the fileset.

Parameters:

`fileset`**:** A physiol_cip_traceset_fileset object.

`traceset_index`**:** Index of traceset item in this fileset (corresponds to row in cells_filename) to find the cell information.

`trace_index`**:** Index of item in the traceset.

`a_db`**:** A DB created by this fileset to read the traceset and item indices from.

`props`**:** A structure with any optional properties, passed to physiol_cip_traceset/cip_trace.

Returns:

a_cip_trace: One or more cip_trace object that holds the raw data.

See also: `loadItemProfile` (p. **??**), `physiol_cip_traceset/cip_trace` (p. 150)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/07/13

A.25.10 Method `physiol_cip_traceset_fileset/loadItemProfile`

Summary: Loads a cip_trace_profile object from a raw data file in the fileset.

Usage:

```
a_profile = loadItemProfile(fileset, traceset_index, trace_index)
```

Parameters:

`fileset`**:** A physiol_cip_traceset object.

`traceset_index` **:** Index of traceset item in this fileset (corresponds to row in cells_filename) to use grab the cell information.

`trace_index`**:** Index of item in the traceset.

Returns:

a_profile: A profile object that implements the getResults method.

See also: `itemResultsRow` (p. **??**), `params_tests_fileset` (p. 135), `paramNames` (p. **??**), `testNames` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14 and Tom Sangrey

---

A.25.11 Method `physiol_cip_traceset_fileset/neuronNameFromId`

Summary: Returns string neuron names from a list of neuron ids.

Usage:

```
name_strs = neuronNameFromId(fileset, neuron_ids)
```

Parameters:

`fileset`: A physiol_cip_traceset_fileset object.

`neuron_ids`: One or more neuron ids in a vector.

Returns:

name_strs: Cell array of neuron names corresponding to the ids given.

See also: `physiol_cip_traceset_fileset` (p. 152)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/11/16

A.26 Class `plot_abstract`

A.26.1 Constructor `plot_abstract/plot_abstract`

Summary: A plot that can be directly visualized or included in subplots.

Usage:

```
obj = plot_abstract(data, axis_labels, title, legend, command, props)
```

Description: Base class that holds the necessary data to draw a plot. This data can then be used to generate different plots. Subclasses define specific plots with additional data. Subclasses should conform to the standard that the series of commands found in plotFigure should produce a valid figure.

Parameters:

`data`: A cell array of data arrays (x, y, z, etc.) that can be
fed to plot commands.

`axis_labels`: Cell array of axis label strings.

`title`: Plot description string.

`legend`: Cell array of descriptions for each item plotted.

`command`: Plotting command to use (Optional, default='plot')

`props`: A structure with any optional properties.

`axisLimits`: Sets axis limits of non-NaN values in vector.

`tightLimits`: If 1, issues an "axis tight" command (default=0)

`border`**:** Relative size of border spacing around axis, between 0 - 1. (default=0)
If a scalar, equal border on all sides, give a four-element vector [left bottom right top] to define borders for each side.

`colormap`**:** Figure colormap passed to the colormap function. If function handle, its output is passed instead.

`grid`**:** Display dashed grid in background.

`noXLabel`**:** No X-axis label.

`noYLabel`**:** No Y-axis label.

`noTitle`**:** No title.

`rotateXLabel`**:** Rotates the X-axis label for smaller width.

`rotateYLabel`**:** Rotates the Y-axis label for smaller width.

`numXTicks`**:** Number of ticks on X-axis.

`formatXTickLabels`**:** The sprintf format string for tick labels.

`XTick, YTick`**:** Point locations for axis ticks.

`XTickLabel, YTickLabel`**:** Axis tick labels.

`ColorOrder`**:** Set the ColorOrder of the axis.

`LineStyleOrder`**:** Set the LineStyleOrder of the axis.

`legendLocation`**:** Passed to legend(..., 'location', legendLocation).

`legendOrientation`**:** Passed to legend(..., 'orientation', legendLocation).

`noLegends`**:** If exists, no legends are displayed.

`axisProps`**:** Passed to set properties of the axis drawn.

`plotProps`**:** Passed to set properties of the plot drawn.

`figureProps`**:** Passed to set properties of the figure drawn.

`PaperPosition`**:** Sets the figure property for printing at this size.

`resizeControl`**:** If 0, drawing after resize is disabled and prints at screen size, if 1 (default), redraws figure after each resize event and prints at PaperPosition size.

`fixedSize`**:** Vector of width and height in inches passed to PaperPosition property. Implies resizeControl=0.

Returns a structure object with the following fields:

data, axis_labels, title, legend, command, props

See also: `plot_abstract/plot` (p. 159), `plot_abstract/plotFigure` (p. 162)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/22

A.26.2 Method `plot_abstract/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.26.3 Method plot_abstract/plot

Summary: Draws this plot in the current axis.

Usage:

```
handles = plot(a_plot, layout_axis)
```

Parameters:

`a_plot`**:** A plot_abstract object, or a subclass object.

`layout_axis`**:** The axis position to layout this plot (Optional). If NaN, doesn't open a new axis.

Returns:

handles: Handles of graphical objects drawn.

See also: `plot_abstract` (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/22

A.26.4 Method plot_abstract/setProp

Summary: Generic method for setting optional object properties.

Usage:

```
obj = setProp(obj, prop1, val1, prop2, val2, ...)
```

Description: Modifies or adds property values. As many property name-value pairs can be specified.

Parameters:

`obj`**:** Any object that has a props field.

`attr`**:** Property name

`val`**:** Property value.

Returns:

obj: The new object with the updated properties.

See also:

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/22

A.26.5 Method `plot_abstract/set`

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.26.6 Method `plot_abstract/subsref`

Summary: Defines generic indexing for objects.

A.26.7 Method `plot_abstract/subsasgn`

Summary: Defines generic index-based assignment for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/02/06

A.26.8 Method `plot_abstract/axis`

Summary: Returns the estimated axis ranges of this plot according to its data.

Usage:

```
ranges = axis(a_plot)
```

Parameters:

`a_plot`**:** A plot_abstract object, or a subclass object.

Returns:

ranges: The ranges as a vector in the same way 'axis' would return.

See also: `plot_abstract` (p. 157), `plot_abstract/plot` (p. 159)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/13

A.26.9 Method `plot_abstract/matrixPlots`

Summary: Superpose multiple plots with common command onto a single axis.

Usage:

```
a_plot = matrixPlots(plots, axis_labels, title_str, props)
```

Parameters:

`plots`**:** Array of plot_abstract or subclass objects.

`axis_labels`: Cell array of axis label strings (optional, taken from plots).

`title_str`: Plot description string (optional, taken from plots).

`props`: A structure with any optional properties passed to the Y stack_plot.

- `titlesPos, yLabelsPos, yTicksPos`: if specified, passed to the X stack_plots.
- `rotateYLabel`: if specified, passed to the X stack_plots.
- `axisLimits`: if specified, passed to the X stack_plots.
- `goldratio`: try to make the figure in this aspect ratio.
- `width, height`: if specified, make the figure have this many plots in corresponding dimension.

Returns:

a_plot: A plot_abstract object.

See also: `plot_abstract` (p. 157), `plot_abstract/plot` (p. 159), `plot_abstract/plotFigure` (p. 162)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/12/07

A.26.10 Method `plot_abstract/display`

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.26.11 Method `plot_abstract/superposePlots`

Summary: Superpose multiple plots with common command onto a single axis.

Usage:

```
a_plot = superposePlots(plots, axis_labels, title_str, command, props)
```

Description: The plot decoration will be taken from the last plot in the list, with the exception of legend labels.

Parameters:

`plots`: Array of plot_abstract or subclass objects.

`axis_labels`: Cell array of axis label strings (optional, taken from plots).

`title_str`: Plot description string (optional, taken from plots).

`command`: Plotting command to use (optional, taken from plots)

`props`: A structure with any optional properties.

- `noLegends`: If exists, no legends are created.

Returns:

a_plot: A plot_abstract object.

See also: plot_abstract (p. 157), plot_abstract/plot (p. 159), plot_abstract/plotFigure (p. 162)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/23

A.26.12 Method plot_abstract/decorate

Summary: Places decorations (titles, labels, ticks, etc.) on the plot.

Usage:

```
handles = decorate(a_plot)
```

Parameters:

a_plot: A plot_abstract object, or a subclass object.

Returns:

handles: Handles of graphical objects drawn.

See also: plot_abstract (p. 157), plot_abstract/plot (p. 159)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/22

A.26.13 Method plot_abstract/plotFigure

Summary: Draws this plot alone in a new figure window.

Usage:

```
handle = plotFigure(a_plot)
```

Parameters:

a_plot: A plot_abstract object, or a subclass object.

title_str: (Optional) String to append to plot title.

Returns:

handle: Handle of new figure.

See also: plot_abstract (p. 157), plot_abstract/plot (p. 159), plot_abstract/decorate (p. 162)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/22

A.26.14 Method `plot_abstract/openAxis`

Summary: Calculates the extents for the axis of this plot and opens it.

Usage:

```
[axis_handle, layout_axis] = openAxis(a_plot, layout_axis)
```

Parameters:

`a_plot`: A plot_abstract object, or a subclass object.

`layout_axis`: The axis position to layout this plot (Optional). If NaN, doesn't open a new axis.

Returns:

handles: Handles of graphical objects drawn.

See also: `plot_abstract` (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/22

A.27 Class `plot_bars`

A.27.1 Constructor `plot_bars/plot_bars`

Summary: Bar plot with error lines in individual axes for each variable.

Usage:

```
a_plot = plot_bars(mid_vals, lo_vals, hi_vals, n_vals, x_labels, y_labels, ...  title,
axis_limits, props)
```

Description: Subclass of plot_stack. The plot_abstract/plot command can be used to plot this data. Rows of *_vals will create grouped bars, columns will create new axes.

Parameters:

`mid_vals`: Middle points of error bars.

`lo_vals`: Low points of error bars.

`hi_vals`: High points of error bars.

`n_vals`: Number of samples used for the statistic (Optional).

`x_labels, y_labels`: Axis labels for each bar group. Must match with data columns.

`title`: Plot description.

`axis_limits`: If given, all plots contained will have these axis limits.

`props`: A structure with any optional properties.

`dispBarsLines`: Choose between using 'bars' or 'lines' to connect the errorbars.

`dispErrorbars`: If 1, display errorbars for lo_vals and hi_vals deviation from mid_vals
(default=1).

`dispNvals`: If 1, display n_vals on top of each bar (default=1).

`groupValues`: List of within-group labels passed to XTickLabels, instead of just a sequence of numbers.

`truncateDecDigits`: Truncate labels to this many decimal digits.

`barAxisProps`: props passed to plot_abstract objects with bar commands

Returns a structure object with the following fields:

plot_abstract

See also: `plot_abstract` (p. 157), `plot_abstract/plot` (p. 159)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/07

A.27.2 Method `plot_bars/set`

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.28 Class `plot_errorbar`

A.28.1 Constructor `plot_errorbar/plot_errorbar`

Summary: Generic errorbar plot.

Usage:

```
a_plot = plot_errorbar(x_vals, mid_vals, lo_vals, hi_vals, line_spec, axis_labels, title,
legend, props)
```

Description: Subclass of plot_abstract. The plot_abstract/plot command can be used to plot this data. Needed to create this as a separate class to have the axis ranges method to measure the errorbars.

Parameters:

`x_vals`: X coordinates of errorbars.

`mid_vals`: Middle points of error bars.

`lo_vals`: Low points of error bars.

`hi_vals`: High points of error bars.

`line_spec`**:** Plot line spec to be passed to errorbar

`axis_labels`**:** Cell array for X, Y axis labels.

`title`**:** Plot description.

`legend`**:** For multiple errorbar plots (matrix form), description of each plot.

`props`**:** A structure with any optional properties to be passed to plot_abstract.

Returns a structure object with the following fields:

plot_abstract.

See also: `plot_abstract` (p. 157), `plot_abstract/plot` (p. 159)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/07

A.28.2 Method `plot_errorbar/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.28.3 Method `plot_errorbar/axis`

Summary: Returns the estimated axis ranges of this plot according to its data.

Usage:

```
ranges = axis(a_plot)
```

Parameters:

`a_plot`**:** A plot_abstract object, or a subclass object.

Returns:

ranges: The ranges as a vector in the same way 'axis' would return.

See also: `plot_abstract` (p. 157), `plot_abstract/plot` (p. 159)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/13

A.29 Class plot_errorbars

A.29.1 Constructor plot_errorbars/plot_errorbars

Summary: Plots distributions of variables with errorbars in separate axes.

Usage:

```
a_plot = plot_errorbars(labels, mid_vals, lo_vals, hi_vals, labels, title, axis_limits,
props)
```

Description: Subclass of plot_stack. The plot_abstract/plot command can be used to plot this data. Each of mid_vals, lo_vals, and hi_vals plot its rows in the same axis and columns in different axes.

Parameters:

`labels`**:** Labels of parameters to appear at bottom of each errorbar.

`mid_vals`**:** Middle points of error bars.

`lo_vals`**:** Low points of error bars.

`hi_vals`**:** High points of error bars.

`title`**:** Plot description.

`axis_limits`**:** If given, all plots contained will have these axis limits.

`props`**:** A structure with any optional properties.

Returns a structure object with the following fields:

plot_abstract, labels.

See also: `plot_abstract` (p. 157), `plot_abstract/plot` (p. 159)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/07

A.30 Class plot_image

A.30.1 Constructor plot_image/plot_image

Summary: Generic image plot.

Usage:

```
a_plot = plot_image(image_data, axis_labels, colorbar_label, title_str, props)
```

Description: Subclass of plot_abstract. The plot_abstract/plot command can be used to plot this data. Needed to create this as a separate class to have the axis ranges method implemented and take advantage of plot_abstract props.

Parameters:

`image_data`**:** 2D matrix with image data.

`axis_labels`**:** Cell array for X, Y axis labels.

`colorbar_label`**:** String to appear next to colorbar.

`title_str`**:** Plot description.

`props`**:** A structure with any optional properties.

`colorbar`**:** If defined, show colorbar next to plot.

`numGrads`**:** Number of poles in the colormap gradient. If 1 (default), it will be a monocolor gradient (e.g., gray-level); if 2, it will be a dual-color gradient (e.g., blue to red) with a black crossing point. This point is determined by the minValue (below). Default numGrads=2, if negative values exist in image_data after scaling.

`minValue,maxValue`**:** Use these value to scale the data by (image_data - minValue) / (maxValue - minValue). Otherwise, its min and max is used.

`colormap`**:** Colormap vector, function name or handle to colormap (e.g., 'jet').

`numColors`**:** Number of colors in colormap. (Rest passed to plotImage.)

Returns a structure object with the following fields:

plot_abstract.

Example:

```
» plotFigure(plot_image(rand(5), 'r1', 'r2', 'rand', 'random matrix'))
```

See also: `plot_abstract` (p. 157), `plot_abstract/plot` (p. 159)

Author: Cengiz Gunay <cgunay@emory.edu>, 2008/04/15

A.30.2 Method `plot_image/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.30.3 Method `plot_image/axis`

Summary: Returns the estimated axis ranges of this plot according to its data.

Usage:

```
ranges = axis(a_plot)
```

Parameters:

`a_plot`: A plot_abstract, or subclass, object.

Returns:

ranges: The ranges as a vector in the same way 'axis' would return.

See also: `plot_abstract` (p. 157), `plot_abstract/plot` (p. 159), `axis` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/13

A.31 Class `plot_inset`

A.31.1 Constructor `plot_inset/plot_inset`

Summary: Superpose multiple plots with individual axis at arbitrary locations.

Usage:

```
a_plot = plot_inset(plots, axis_locations, title_str, props)
```

Description: Subclass of plot_abstract. Contains other plot_abstract objects or subclasses thereof to be layout in arbitaray format. Allows overlapping and therefore good for insets and special plots.

Parameters:

`plots`: Cell array of plot_abstract or subclass objects.

`axis_locations`: Matrix of four-element vectors for each given plot.

`title_str`: Title to go on top of the stack

`props`: A structure with any optional properties.

`positioning`: axis_locations interpreted as 'absolute' values or 'relative' to the 1st plot (default='absolute'). Relative positioning doesn't work well.

Returns a structure object with the following fields:

plot_abstract, plots, axis_locations.

See also: `plot_abstract` (p. 157), `plot_abstract/plotFigure` (p. 162)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/06/05

A.31.2 Method `plot_inset/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.31.3 Method `plot_inset/plot`

Summary: Superposes contained plots in their own axes.

Usage:

```
handles = plot(a_plot, layout_axis)
```

Parameters:

`a_plot`**:** A plot_superpose object.

`layout_axis`**:** The axis position to layout this plot (Optional).

Returns:

handles: Handles of graphical objects drawn.

See also: `plot_abstract` (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/06/08

A.31.4 Method `plot_inset/set`

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.32 Class `plot_simple`

A.32.1 Constructor `plot_simple/plot_simple`

Summary: Abstract description of a single plot.

Usage:

```
a_plot = plot_simple(data_x, data_y, title, label_x, label_y, legend, command, props)
```

Description: Subclass of plot_abstract. The plot_abstract/plot command can be used to plot this data.

Parameters:

`data_x`**:** X-axis values for the plot.

`data_y`**:** Y-axis values for the plot.

`title`**:** Plot description.

`label_x`**:** X-axis label string.

`label_y`**:** Y-axis label string.

`legend`**:** Short description of data points.

`command`: Plotting command to use (Optional, default='plot')

`props`: A structure with any optional properties.

Returns a structure object with the following fields:

plot_abstract.

See also: `plot_abstract` (p. 157), `plot_abstract/plot` (p. 159)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/22

A.32.2 Method `plot_simple/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.32.3 Method `plot_simple/set`

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.33 Class `plot_stack`

A.33.1 Constructor `plot_stack/plot_stack`

Summary: A horizontal or vertical stack of plots.

Usage:

```
a_plot = plot_stack(plots, axis_limits, orientation, title_str, props)
```

Description: Subclass of plot_abstract. Contains other plot_abstract objects or subclasses thereof to be layout in stack format.

Parameters:

`plots`: Cell array of plot_abstract or subclass objects.

`axis_limits`: If given, all plots contained will have these axis limits. In this vector, NaNs are untouched, Infs are replaced by minimal and maximal ranges of the stacked plots.

`orientation`: Stack orientation 'x' for horizontal, 'y' for vertical, etc.

`title_str`: Title to go on top of the stack

`props`: A structure with any optional properties.

`yLabelsPos`: 'left' means only put y-axis label to leftmost plot.

`yTicksPos`: 'left' means only put y-axis ticks to leftmost plot.

`xLabelsPos`: 'bottom' means only put x-axis label to lowest plot.

`xTicksPos`: 'bottom' means only put x-axis ticks to lowest plot.

`titlesPos`: 'top' means only put title to top plot.

`relaxedLimits`: Add 10

`relativeSizes`: An array specifying relative size of each plot with one value.
(Example: relativeSizes=[1 2] makes second plot twice wider than first.)

Returns a structure object with the following fields:

plot_abstract, plots, axis_limits, orient.

See also: `plot_abstract` (p. 157), `plot_abstract/plotFigure` (p. 162)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/04

A.33.2 Method `plot_stack/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.33.3 Method `plot_stack/plot`

Summary: Draws this plot in the current axis or at the position in layout_axis.

Usage:

```
handles = plot(a_plot, layout_axis)
```

Parameters:

`a_plot`: A plot_abstract object, or a subclass object.

`layout_axis`: The axis position to layout this plot (Optional).

Returns:

handles: Handles of graphical objects drawn.

See also: `plot_stack` (p. 170), `plot_abstract` (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/04

A.33.4 Method `plot_stack/set`

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.33.5 Method `plot_stack/display`

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.33.6 Method `plot_stack/superposePlots`

Summary: Superpose multiple plot_stack objects that contain exact same contents.

Usage:

```
a_plot = superposePlots(plots, axis_labels, title_str, command, props)
```

Description: The plot decoration will be taken from the last plot in the list, with the exception of legend labels.

Parameters:

- `plots`**:** Array of plot_stack objects.
- `axis_labels`**:** Cell array of axis label strings (optional, taken from plots).
- `title_str`**:** Plot description string (optional, taken from plots).
- `command`**:** Plotting command to use (optional, taken from plots)
- `props`**:** A structure with any optional properties.
  - `noLegends`**:** If exists, no legends are created.

Returns:

a_plot: A plot_stack object.

See also: `plot_abstract` (p. 157), `plot_abstract/plot` (p. 159), `plot_abstract/plotFigure` (p. 162)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/06/14

A.33.7 Method plot_stack/decorate

Summary: No additional decorations for stacked plots.

Usage:

```
a_histogram_db = decorate(a_plot)
```

Parameters:

`a_plot`: A plot_abstract object, or a subclass object.

Returns:

handles: Handles of graphical objects drawn.

See also: plot_abstract (p. 157), plot_abstract/plot (p. 159)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/04

A.34 Class plot_superpose

A.34.1 Constructor plot_superpose/plot_superpose

Summary: Multiple plot_abstract objects superposed on the same axis.

Usage:

```
obj = plot_superpose(plots, axis_labels, title_str, props)
```

Description: Subclass of plot_abstract. Contains multiple plot_abstract objects to be plotted on the same axis. This is different than the plot_abstract/superpose, where only using the same plot command is allowed. Here, each plot_abstract can have its own special plotting command. Subclasses of plot_abstract is also allowed here. The decorations comes from this object and not children plots. This behavior is different than plot_stack, where each plot has its own decorations. If you want each plot to have its own axis (e.g. an inset, or plot with multiple axis labels) then you should use plot_inset.

Parameters:

`plots`: Cell array of plot_abstract or subclass objects.

`axis_labels`: Cell array of axis label strings.

`title_str`: Plot description string.

`props`: A structure with any optional properties (passed to plot_abstract).

Returns a structure object with the following fields:

plot_abstract, plots

See also: plot_abstract/superpose (p. **??**), plot_superpose/plot (p. 174)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/22

A.34.2 Method plot_superpose/get

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.34.3 Method plot_superpose/plot

Summary: Draws this plot in the current axis.

Usage:

```
handles = plot(a_plot, layout_axis)
```

Parameters:

a_plot: A plot_superpose object.
layout_axis: The axis position to layout this plot (Optional).

Returns:

handles: Handles of graphical objects drawn.

See also: plot_abstract (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/04/08

A.34.4 Method plot_superpose/set

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.34.5 Method plot_superpose/axis

Summary: Returns the maximal axis ranges according to superposed subplots.

Usage:

```
ranges = axis(a_plot)
```

Parameters:

a_plot: A plot_abstract object, or a subclass object.

Returns:

ranges: The ranges as a vector in the same way 'axis' would return.

See also: plot_abstract (p. 157), plot_abstract/plot (p. 159)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/05/22

A.34.6 Method `plot_superpose/display`

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.34.7 Method `plot_superpose/superposePlots`

Summary: Superpose multiple plot_superpose objects by merging them into one.

Usage:

```
a_plot = superposePlots(plots, axis_labels, title_str, command, props)
```

Parameters:

`plots`**:** Array of plot_superpose objects.

`axis_labels`**:** Cell array of axis label strings (optional, taken from plots).

`title_str`**:** Plot description string (optional, taken from plots).

`command`**:** Plotting command to use (optional, taken from plots)

`props`**:** A structure with any optional properties.

`noLegends`**:** If exists, no legends are created.

Returns:

a_plot: A plot_superpose object.

See also: `plot_abstract/superposePlots` (p. 161), `plot_stack/superposePlots` (p. 172)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/06/14

A.34.8 Method `plot_superpose/decorate`

Summary: Places decorations using the first plot of the superposed plots.

Usage:

```
handles = decorate(a_plot)
```

Parameters:

`a_plot`**:** A plot_abstract object, or a subclass object.

Returns:

handles: Handles of graphical objects drawn.

See also: `plot_abstract` (p. 157), `plot_abstract/plot` (p. 159)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/04/11

A.35 Class ranked_db

A.35.1 Constructor ranked_db/ranked_db

Summary: A database of distance values generated by ranking rows of orig_db with the criterion in crit_db.

Usage:

```
a_ranked_db = ranked_db(data, col_names, orig_db, crit_db, id, props)
```

Description: This is a subclass of tests_db. It should contain a Distance column. A more general ranked db class may be needed later. Use the rankMatching method to get an instance of this class.

Parameters:

`data`**:** Database contents.

`col_names`**:** The column names.

`orig_db`**:** DB whose rows are ranked.

`crit_db`**:** The criterion DB used for generating the ranking scores.

`id`**:** An identifying string.

`props`**:** A structure with any optional properties.

`tolerateNaNs`**:** If 0, rows with any NaN values are skipped (default=1).

Returns a structure object with the following fields:

tests_db, orig_db, crit_db, props.

See also: `tests_db` (p. 227), `tests_db/rankMatching` (p. 249), `tests_db/matchingRow` (p. 229)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/12/21

A.35.2 Method ranked_db/get

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.35.3 Method ranked_db/plotRowErrors

Summary: Create plot of rankings with errors associated with each measure color-coded.

Usage:

```
a_plot = plotRowErrors(a_ranked_db, rows, props)
```

Parameters:

`a_ranked_db`**:** A ranked_db object.

`rows`**:** Indices of rows in a_ranked_db.

`title_str`**:** (Optional) String to append to plot title.

`props`**:** A structure with any optional properties.

- `sortMeasures`**:** If specified, measure order is determined with increasing overall distance.
- `RowName`**:** Label to show on X-axis (default='Ranks')
- `rowSteps`**:** Steps to jump in labeling rows on the x-axis.
- `superposeDistances`**:** Superpose a white-colored distance line plot. (rest passed to plot_abstract)

Returns:

a_plot: A plot_abstract object.

See also: `ranked_db` (p. 176), `tests_db/rankMatching` (p. 249), `plot_abstract` (p. 157), `plotImage` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/12/12

A.35.4 Method ranked_db/joinOriginal

Summary: Joins the distance values to the original db rows with matching row indices.

Usage:

```
a_db = joinOriginal(a_ranked_db, rows)
```

Description: Takes the parameter columns from orig_db and all tests from crit_db.

Parameters:

`a_ranked_db`**:** A ranked_db object.

`rows`**:** Join only the given rows.

Returns:

a_db: A params_tests_db object (same type as a_ranked_db.orig_db) containing the desired rows in ascending order of distance.

See also: `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/12/21

A.35.5 Method `ranked_db/plotDistMatrix`

Summary: Create a color-coded matrix plot of with total errors from the ranked DB.

Usage:

```
a_plot = plotDistMatrix(db, rows, col_size, col_name, num_col_labels, row_name,
num_row_labels, title_str, props)
```

Description: The col_size parameter is used to find the number of rows that make up the x-dimension of the color matrix plot.

Parameters:

`db`: A ranked_db object.

`rows`: Indices of rows in db after joining (and sorting).

`col_size`: Number of rows to take from DB to form the columns of matrix plot.

`col_name, row_name`: DB column to use for the figure column and row, respectively.

`num_col_labels, num_row_labels`: Number of labels to put on each axis.

`title_str`: If non-empty, replaces generic title with db name.

`props`: A structure with any optional properties.

`sortBy`: If specified, db is sorted after being joined with original using this column.

`colorbar`: Put a colorbar on the figure.
(also passed to plot_abstract)

Returns:

a_plot: A plot_abstract object.

Example:

```
» plotFigure(plotDistMatrix(scored_blocked_sk_gps0503b_control_db, ':', 10, 'SK', 10,
'trial', 10, 'gps0503b (control), preset 6 - top 50 matches', struct('sortBy', 'trial',
'colorbar', 1, 'PaperPosition', [0 0 5 3])));
```

See also: `ranked_db` (p. 176), `plot_abstract` (p. 157), `getDistMatrix` (p. **??**), `plotCompareDistMatx` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/12/12

A.35.6 Method `ranked_db/set`

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.35.7 Method `ranked_db/renameColumns`

Summary: Rename an existing column or columns.

Usage:

```
a_db = renameColumns(a_db, test_names, new_names)
```

Description: This method is an overloaded method for ranked_db that keeps consistent the column names of the ranked, criterion and original DBs. The other DBs are not renamed for the Distance and RowIndex columns.

Parameters:

`a_db`**:** A ranked_db object.

`test_names`**:** A cell array of existing test names.

`new_names`**:** New names to replace existing ones.

Returns:

a_db: The ranked_db object that includes the new columns.

See also: `tests_db/renameColumns` (p. 251)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/06/07

A.35.8 Method `ranked_db/subsref`

Summary: Defines generic indexing for objects.

A.35.9 Method `ranked_db/plotCompareDistMatx`

Summary: Compare differences and correlations of distance matrices from two ranked DBs.

Usage:

```
a_plot = plotCompareDistMatx(db, rows, col_size, col_name, num_col_labels, row_name,
num_row_labels, title_str, props)
```

Description: Produces three plots: (1) distance difference matrix, (2) 2D cross-correlogram, and (3) repeated 1D cross-correlogram for each row.

Parameters:

`db, w_db`**:** The ranked_db objects to be compared.

`rows`**:** Indices of rows in db after joining (and sorting) for both DBs.

`col_size`**:** Number of rows to take from DB to form the columns of matrix plot.

`col_name, row_name`**:** DB column to use fot the figure column and row, respectively.

`num_col_labels, num_row_labels`**:** Number of labels to put on each axis.

`title_str`**:** If non-empty, replaces generic title with db name.

`props`**:** A structure with any optional properties.

> `sortBy`**:** If specified, db is sorted after being joined with original using this column.
>
> `colorbar`**:** Put a colorbar on the figure.
> (also passed to plot_abstract)

Returns:

a_plot: A plot_abstract object.

See also: `tests_db` (p. 227), `plot_abstract` (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/12/12

A.35.10 Method `ranked_db/displayRows`

Summary: Displays rows of rankings together with errors associated with each measure.

Usage:

```
s = displayRows(db, rows)
```

Parameters:

`db`: A tests_db object.

`rows`: Indices of rows in db.

Returns:

s: A structure of column name and value pairs.

See also: `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/15

A.35.11 Method `ranked_db/getDistMatrix`

Summary: Create a matrix of total errors from the ranked DB.

Usage:

```
distmatx = getDistMatrix(db, rows, col_size, props)
```

Description: The col_size parameter is used to find the number of rows that make up the x-dimension of the matrix.

Parameters:

`db`: A tests_db object.

`rows`: Indices of rows in db after joining (and sorting).

`col_size`: Number of rows to take from DB to form the columns of matrix plot.

`props`: A structure with any optional properties.

`sortBy`: If specified, db is sorted after being joined with original using this column.

Returns:

a_plot: A plot_abstract object.

See also: `tests_db` (p. 227), `plot_abstract` (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/12/12

A.35.12 Method ranked_db/blockedDistances

Summary: Creates a db of distances to blocked versions of top ranks.

Usage:

```
[a_db, ranked_dbs] = blockedDistances(a_ranked_db, rows, blocked_db, blocked_param_indices,
block_levels, crit_db)
```

Parameters:

`a_ranked_db`**:** A ranked_db object.

`rows`**:** Use the given row rankings.

`blocked_db`**:** db with blocked versions of original ranks.

`blocked_param_indices`**:** Indices of parameters to be blocked.

`block_levels`**:** Number of parameter levels for blocking.

`crit_db`**:** Calculate distance from this criterion.

Returns:

a_db: A tests_db object with the matrix of distances. ranked_dbs: A cell array of ranked_dbs for each row.

Example:

```
» dist_matx_db = blockedDistances(rankMatching(super_db, matchingRow(rsuper_phys_db, 20)),
1:5, super_blocker_db, [1 2], 10, matchingRow(rsuper_phys_db, 21))
```

See also: `makeModifiedParamDB` (p. **??**), `getParamRowIndices` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/01/14

A.36 Class results_profile

A.36.1 Constructor results_profile/results_profile

Summary: Creates and collects result profiles for data objects.

Usage:

```
obj = results_profile(results, id, props)
```

Description: This is the base class for all profile classes.

Parameters:

`results`**:** A structure containing test results.

`id`**:** Identification string.

`props`**:** A structure with any optional properties.

Returns a structure object with the following fields:

results, id, props.

See also: `trace_profile` (p. 288), `cip_trace_profile` (p. 71)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.36.2 Method `results_profile/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.36.3 Method `results_profile/plot`

Summary: Generic method to plot a tests_db or a subclass. Requires a plot_abstract method to be defined for this object.

Usage:

```
h = plot(a_tests_db, title_str)
```

Parameters:

`a_tests_db`**:** A histogram_db object.

`title_str`**:** (Optional) String to append to plot title.

Returns:

h: The figure handle created.

See also: `plot_abstract` (p. 157), `plotFigure` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/06

A.36.4 Method `results_profile/getResults`

Summary: Return the results profile structure.

Usage:

```
results = getResults(p)
```

Parameters:

`p`**:** A result_profile object.

Returns:

results: A structure associating test names to values.

See also: `results_profile` (p. 182)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.36.5 Method `results_profile/subsref`

Summary: Defines generic indexing for objects.

A.36.6 Method `results_profile/display`

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.37 Class `script_array`

A.37.1 Constructor `script_array/script_array`

Summary: Generic class that provides the scripts for a repetitive array job.

Usage:

```
obj = script_array(num_runs, id, props)
```

Description: This is the base class for all script_array classes. Runs the runJob method as num_runs many times.

Parameters:

`num_runs`: The number of times the runJob script should be evoked.

`id`: Identification string.

`props`: A structure with any optional properties.

`runJobFunc`: A function name or handle to be used instead of default runJob.

Returns a structure object with the following fields:

num_runs, id, props.

Example:

```
» func1 = inline('x^2')
» runFirst(script_array(10, 'squares numbers up to 10'), struct('runJobFunc', func1))
ans = [ 1] [ 4] [ 9] [ 16] [ 25] [ 36] [ 49] [ 64] [ 81] [100]
```

See also: `runFirst` (p. **??**), `runLast` (p. **??**), `runJob` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/02/01

A.37.2 Method script_array/get

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.37.3 Method script_array/runJob

Summary: Method to be called for each of the script_array jobs.

Usage:

```
job_result = runJob(a_script_array, vector_index)
```

Description: This method is provided as a placeholder and does nothing. If the run_job_func property is defined, it will call that function.

Parameters:

a_script_array: A script_array object.

vector_index: The index within the vector job.

Returns:

job_result: Any output produced by the job.

Example:

```
See real example in script_array.  Call the 5th job:
» runJob(script_array(10, 'this one does nothing for 10 times'), 5);
```

See also: runLast (p. **??**), runFirst (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/02/01

A.37.4 Method script_array/runLast

Summary: Method to be called last after the script_array jobs.

Usage:

```
job_results = runLast(a_script_array, job_results)
```

Description: This method is provided as a placeholder and does nothing. It can filter-out the results returned from the jobs run. Normally it is invoked internally by the runFirst method, after running and collecting results from the vector jobs with the runJob method.

Parameters:

a_script_array: A script_array object.

`job_results`: The index within the vector job.

Returns:

job_results: Any output produced by the job.

Example:

```
Call it directly:
» runLast(script_array(10, 'this one does nothing for 10 times'), );
```

See also: `runJob` (p. **??**), `runFirst` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/02/01

A.37.5 Method `script_array/set`

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/02/06

A.37.6 Method `script_array/subsref`

Summary: Defines generic indexing for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.37.7 Method `script_array/subsasgn`

Summary: Defines generic index-based assignment for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/02/06

A.37.8 Method `script_array/runFirst`

Summary: Method to be called at beginning of script_array jobs.

Usage:

```
job_results = runFirst(a_script_array)
```

Description: This method initiates the script_array jobs. It loops and calls runJob and finally calls runLast.

Parameters:

`a_script_array`: A script_array object.

Returns:

job_results: A cell array of results collected from each item of the vector jobs.

Example:

```
» runFirst(script_array(10, 'this one does nothing for 10 times'));
```

See also: runLast (p. **??**), runJob (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/02/01

A.38 Class script_array_for_cluster

A.38.1 Constructor script_array_for_cluster/script_array_for_cluster

Summary: Generic class defining a repetitive vector job to be run on a Sun Grid Engine (SGE) computing cluster.

Usage:

```
a_script_cluster = script_array_for_cluster(num_runs, sge_wrapper_script, id, props)
```

Description: This is a subclass of the script_array class. The runFirst method spawns num_runs copies of the runJob method in parallel on the cluster, followed by the invocation of the runLast method.

Parameters:

`num_runs`**:** The number of times the runJob script should be evoked.

`sge_wrapper_script`**:** A script that can be submitted with qsub and can execute arbitrary

Matlab commands on the cluster nodes. It can have qsub options prepended to it such as '-p -100 -q all.q <abs_path_to>/sge_matlab.sh'.

`id`**:** Identification string.

`props`**:** A structure with any optional properties.

`notifyByMail`**:** An SGE notification email is sent to this address after lastJob.
(others passed to script_array)

Returns a structure object with the following fields:

num_runs, id, props.

See also: runFirst (p. **??**), runLast (p. **??**), runJob (p. **??**), script_array (p. 184)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/02/02

A.38.2 Method script_array_for_cluster/get

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.38.3 Method script_array_for_cluster/set

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/02/06

A.38.4 Method script_array_for_cluster/runFirst

Summary: Method to be called at beginning of script_array_for_cluster jobs.

Usage:

```
job_results = runFirst(a_script_cluster)
```

Description: This method initiates the script_array_for_cluster jobs. It submits an SGE vector job for running each runJob and finally runLast. There is no way of collecting outputs from individual runJob calls.

Parameters:

a_script_cluster: A script_array_for_cluster object.

Returns:

job_results: A cell array of results collected from each item of the vector jobs.

Example:

```
» runFirst(script_array_for_cluster(10, 'this one does nothing for 10 times'));
```

See also: script_array_for_cluster (p. 187)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/02/01

A.39 Class `script_factory`

A.39.1 Constructor `script_factory/script_factory`

Summary: Generic class to automatically create a set of scripts.

Usage:

```
obj = script_factory(num_scripts, out_name, id, props)
```

Description: This is the base class for all script_factory classes.

Parameters:

- `num_scripts`: Number of scripts to create.
- `out_name`: The file name for the output scripts. A ’ filename corresponds to the script number.
- `id`: Identification string.
- `props`: A structure with any optional properties.

Returns a structure object with the following fields:

num_scripts, out_name, id, props.

See also: `script_factory/writeScripts` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/11/28

A.39.2 Method `script_factory/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.40 Class `spike_shape`

A.40.1 Constructor `spike_shape/spike_shape`

Summary: An action potential shape trace.

Usage:

```
obj = spike_shape(data, dt, dy, id)
```

Parameters:

- `data`: A vector of data points containing the spike shape.
- `dt`: Time resolution [s].
- `dy`: y-axis resolution [ISI (V, A, etc.)]

`id`: Identification string.

`props`: A structure with any optional properties.

`baseline`: Resting potential.

`threshold`: Spike threshold.

`init_Vm_method`: Method to obtain spike initiation voltage.
1- maximum acceleration point 2- threshold crossing of acceleration (needs threshold) 3- threshold crossing of slope (needs threshold) 4- maximum acceleration in phase space (optionally specify maximal threshold as init_threshold) 5- point of maximum curvature, when slope is between init_lo_thr and init_hi_thr 6- local maximum of second derivative in the phase space nearest slope crossing init_threshold 7- threshold crossing of interpolated slope (needs threshold) 8- maximum curvature in phase-plane 9- Combined curvature and inflection method in time-domain.

`init_threshold`: Spike initiation threshold (deriv or accel).
(see above methods and implementation in calcInitVm)

`init_lo_thr, init_hi_thr`: Low and high thresholds for slope.

Returns a structure object with the following fields:

trace, props.

See also: `trace/spike_shape` (p. 288), `trace/analyzeSpikesInPeriod` (p. 282), `trace` (p. 278), `spikes` (p. 202), `period` (p. 140)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/07/30

A.40.2 Method `spike_shape/calcInitVmV2PPLocal`

Summary: Calculates the action potential threshold by finding the local second derivative maximum in voltage-time slope versus voltage phase plane, nearest a slope threshold crossing.

Usage:

```
[init_idx, a_plot] = calcInitVmV2PPLocal(s, max_idx, min_idx, lo_thr, plotit)
```

Parameters:

`s`: A spike_shape object.

`max_idx`: The index of the maximal point of the spike_shape [dt].

`min_idx`: The index of the minimal point of the spike_shape [dt].

`lo_thr`: Lower threshold for time voltage slope.

`plotit`: If non-zero, plot a graph annotating the test results (optional).

Returns:

init_idx: Its index in the spike_shape [dt]. a_plot: plot_abstract, if requested.

See also: calcInitVm (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/18

A.40.3 Method spike_shape/get

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.40.4 Method spike_shape/calcMinVm

Summary: Calculates the minimal value of the spike_shape, s.

Usage:

```
[min_val, min_idx, max_min_time] = calcMinVm(s, max_idx)
```

Parameters:

s: A spike_shape object.

max_idx: The index of the maximal point of the spike_shape [dt].

Returns:

min_val: The min value [dy]. min_idx: Its index in the spike_shape [dt]. max_min_time: Time from max to min [dt].

See also: period (p. 140), spike_shape (p. 189), trace/calcMin (p. 284)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/02

A.40.5 Method spike_shape/calcInitVmSekerliV2

Summary: Calculates the action potential threshold using the maximum second derivative of the phase space of voltage-time slope versus voltage.

Usage:

```
[init_idx, a_plot] = calcInitVmSekerliV2(s, max_idx, min_idx, plotit)
```

Parameters:

s: A spike_shape object.

`max_idx`: The index of the maximal point of the spike_shape [dt].

`min_idx`: The index of the minimal point of the spike_shape [dt].

`plotit`: If non-zero, plot a graph annotating the test results (optional).

Returns:

init_idx: Its index in the spike_shape [dt]. a_plot: plot_abstract, if requested.

See also: `calcInitVm` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/18

A.40.6 Method `spike_shape/calcInitVmMaxCurvature`

Summary: Calculates the action potential threshold using the maximum of the curvature equation.

Usage:

```
[init_idx, a_plot] = calcInitVmMaxCurvature(s, max_idx, min_idx, plotit)
```

Description: Point of maximum curvature: Kp = V”[1 + (V’)^2]^(-3/2) Taken from Sekerli, Del Negro, Lee and Butera. IEEE Trans. Biomed. Eng., 51(9): 1665-71, 2004.

Parameters:

`s`: A spike_shape object.

`max_idx`: The index of the maximal point of the spike_shape [dt].

`min_idx`: The index of the minimal point of the spike_shape [dt].

`plotit`: If non-zero, plot a graph annotating the test results (optional).

Returns:

init_idx: AP threshold index in the spike_shape [dt]. a_plot: plot_abstract, if requested.

See also: `calcInitVm` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/19

A.40.7 Method `spike_shape/calcMaxVm`

Summary: Calculates the maximal value of the spike_shape, s.

Usage:

```
[max_val, max_idx] = calcMaxVm(s)
```

Parameters:

`s`**:** A spike_shape object.

Returns:

max_val: The max value. max_idx: Its index in the spike_shape [dt].

See also: `period` (p. 140), `spike_shape` (p. 189), `trace/calcMax` (p. 281)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/02

A.40.8 Method `spike_shape/getResults`

Summary: Runs all tests defined by this class and return them in a structure.

Usage:

```
[results, a_plot] = getResults(s, plotit)
```

Parameters:

`s`**:** A spike_shape object.

`plotit`**:** If non-zero, plot a graph annotating the test results (optional).

Returns:

results: A structure associating test names to values in ms and mV. a_plot: plot_abstract, if requested.

See also: `spike_shape` (p. 189)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/02

A.40.9 Method spike_shape/plotCompareMethodsSimple

Summary: Creates a multi-plot comparing different action potential threshold finding methods.

Usage:

```
a_plot = plotCompareMethodsSimple(s, title_str)
```

Parameters:

s: A spike_shape object.

title_str: Title suffix (optional).

Returns:

a_plot: A plot_abstract object that can be visualized.

See also: spike_shape (p. 189), plot_abstract (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/19

A.40.10 Method spike_shape/set

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.40.11 Method spike_shape/plotPP

Summary: Plots the dV/dt vs. V phase-plane representation of the spike shape.

Usage:

```
a_plot = plotPP(s)
```

Parameters:

s: A spike_shape object.

Returns:

a_plot: A plot_abstract object that can be visualized.

See also: spike_shape (p. 189), plot_abstract (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/16

A.40.12 Method spike_shape/calcInitVmSlopeThresholdSupsample

Summary: Estimates the AP threshold as the first slope threshold crossing by first supersampling the data using cubic spline interpolation.

Usage:

```
[init_idx, a_plot] = calcInitVmSlopeThresholdSupsample(s, max_idx, min_idx, thr, plotit)
```

Parameters:

s: A spike_shape object.

max_idx: The index of the maximal point of the spike_shape [dt].

min_idx: The index of the minimal point of the spike_shape [dt].

thr: Threshold for time derivative of voltage.

plotit: If non-zero, plot a graph annotating the test results (optional).

Returns:

init_idx: AP threshold index in the spike_shape [dt]. a_plot: plot_abstract, if requested.

See also: calcInitVm (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/03/23

A.40.13 Method spike_shape/calcInitVmV3hKpTinterp

Summary: Calculates candidates for action potential threshold using the first three time-domain derivatives.

Usage:

```
[init_idx, a_plot] = calcInitVmV3hKpTinterp(s, max_idx, min_idx, lo_thr, hi_thr, plotit)
```

Description: First uses interpolation to increase time points. Calculates h, the second derivative of phase-plane (d^2 v'/dv^2), in terms of time-domain derivatives. Also calculates Kp = V"[1 + (V')^2]^(-3/2), the curvature. The maxima of these functions are used as candidates for AP thresholds.

Parameters:

s: A spike_shape object.

max_idx: The index of the maximal point of the spike_shape [dt].

min_idx: The index of the minimal point of the spike_shape [dt].

lo_thr, hi_thr: Lower and higher thresholds for time derivative of voltage.

`plotit`: If non-zero, plot a graph annotating the test results (optional).

Returns:

init_idx: Indices of threshold candidates in the spike_shape [dt]. a_plot: plot_abstract, if requested.

See also: `calcInitVm` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/18

A.40.14 Method `spike_shape/plotCompareMethods`

Summary: Creates a multi-plot comparing different action potential threshold finding methods.

Usage:

```
a_plot = plotCompareMethods(s, title_str)
```

Parameters:

`s`: A spike_shape object.

`title_str`: Title suffix (optional).

Returns:

a_plot: A plot_abstract object that can be visualized.

See also: `spike_shape` (p. 189), `plot_abstract` (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/19

A.40.15 Method `spike_shape/plotResults`

Summary: Plots the spike shape annotated with result characteristics.

Usage:

```
a_plot = plotResults(s, title_str, props)
```

Parameters:

`s`: A spike_shape object.

Returns:

a_plot: A plot_abstract object that can be visualized. title_str: (Optional) String to append to plot title. props: A structure with any optional properties, passed to trace/plotData.

See also: `spike_shape` (p. 189), `plot_abstract` (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/17

A.40.16 Method `spike_shape/display`

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.40.17 Method `spike_shape/calcInitVmLtdMaxCurv`

Summary: Calculates the action potential threshold using the maximum of the curvature equation only in the limited range given with two voltage slope thresholds.

Usage:

```
[init_idx, a_plot] = calcInitVmLtdMaxCurv(s, max_idx, min_idx, lo_thr, hi_thr, plotit)
```

Description: Point of maximum curvature: Kp = V"[1 + (V')^2]^(-3/2) Taken from Sekerli, Del Negro, Lee and Butera. IEEE Trans. Biomed. Eng., 51(9): 1665-71, 2004.

Parameters:

`s`: A spike_shape object.

`max_idx`: The index of the maximal point of the spike_shape [dt].

`min_idx`: The index of the minimal point of the spike_shape [dt].

`lo_thr, hi_thr`: Lower and higher thresholds for time derivative of voltage.

`plotit`: If non-zero, plot a graph annotating the test results (optional).

Returns:

init_idx: AP threshold index in the spike_shape [dt]. a_plot: plot_abstract, if requested.

See also: `calcInitVm` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/19

A.40.18 Method `spike_shape/calcInitVmMaxCurvPhasePlane`

Summary: Calculates the voltage at the maximum curvature in the phase plane as action potential threshold.

Usage:

```
[init_idx, max_d1o, a_plot, fail_cond] = calcInitVmMaxCurvPhasePlane(s, max_idx, min_idx,
plotit)
```

Description: First take the phase-plane v'-v from the beginning to max(v'). Then regulate intervals by interpolation. Point of maximum curvature: Kp = V"[1 + (V')^2]^(-3/2) Taken from Sekerli, Del Negro, Lee and Butera. IEEE Trans. Biomed. Eng., 51(9): 1665-71, 2004.

Parameters:

`s`**:** A spike_shape object.

`max_idx`**:** The index of the maximal point of the spike_shape [dt].

`min_idx`**:** The index of the minimal point of the spike_shape [dt].

`plotit`**:** If non-zero, plot a graph annotating the test results (optional).

Returns:

init_idx: AP threshold index in the spike_shape [dt]. max_d1o: Maximal value of first voltage derivative [dy]. a_plot: plot_abstract, if requested. fail_cond: True if this algorithm fails to be trustable.

See also: `calcInitVm` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/04/12

A.40.19 Method `spike_shape/plotTPP`

Summary: Plots the dV/dt vs. V phase-plane representation of the spike shape.

Usage:

```
a_plot = plotTPP(s)
```

Description: Uses the Taylor series estimation for finding the derivative dV/dt.

Parameters:

`s`**:** A spike_shape object.

Returns:

a_plot: A plot_abstract object that can be visualized.

See also: spike_shape (p. 189), plot_abstract (p. 157), diffT (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/16

A.40.20 Method spike_shape/calcInitVm

Summary: Calculates spike threshold related measures of the spike_shape, s.

Usage:

```
[init_val, init_idx, rise_time, amplitude, peak_mag, peak_idx, max_d1o, a_plot] =
calcInitVm(s, max_idx, min_idx)
```

Parameters:

- s: A spike_shape object.
- max_idx: The index of the maximal point of the spike_shape [dt].
- min_idx: The index of the minimal point of the spike_shape [dt].
- plotit: If non-zero, plot a graph annotating the test results (optional).

Returns:

init_val: The potential value [dy]. init_idx: Its index in the spike_shape [dt]. rise_time: Time from initiation to maximum [dt]. amplitude: Magnitude from initiation to max [dy]. peak_mag: Peak value [dy]. peak_idx: Extrapolated spike peak index [dt]. max_d1o: Maximal value of first voltage derivative [dy]. a_plot: plot_abstract, if requested.

See also: spike_shape (p. 189)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/02

A.40.21 Method spike_shape/calcWidthFall

Summary: Calculates the spike width and fall information of the spike_shape, s.

Usage:

```
[base_width, half_width, half_Vm, fall_time, min_idx, min_val, max_ahp, ahp_decay_constant,
dahp_mag, dahp_idx] = ...  calcWidthFall(s, max_idx, max_val, init_idx, init_val)
```

Description: max_* can be the peak_* from calcInitVm.

Parameters:

- s: A spike_shape object.
- max_idx: The index of the maximal point [dt].

`max_val`: The value of the maximal point [dy].

`init_idx`: The index of spike initiation point [dt].

`init_val`: The value of spike initiation point [dy].

`fixed_Vm`: The desired height for width calculation [V].

Returns:

base_width: Width of spike at base [dt] half_width: Width of spike at half_Vm [dt] half_Vm: Half height of spike [dy] fall_time: Time from peak to initialization level [dt]. min_idx: The index of the minimal point of the spike_shape [dt]. max_ahp: Magnitude from initiation to minimum [dy]. ahp_decay_constant: Approximation to refractory decay after maxAHP [dt]. dahp_mag: Magnitude of the double AHP peak dahp_mag: Index of the double AHP peak

See also: `spike_shape` (p. 189)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/02

A.40.22 Method `spike_shape/calcInitVmSlopeThreshold`

Summary: Calculates the AP threshold using the slope threhold crossing.

Usage:

```
[init_idx, a_plot] = calcInitVmSlopeThreshold(s, max_idx, min_idx, thr, plotit)
```

Parameters:

`s`: A spike_shape object.

`max_idx`: The index of the maximal point of the spike_shape [dt].

`min_idx`: The index of the minimal point of the spike_shape [dt].

`thr`: Threshold for time derivative of voltage.

`plotit`: If non-zero, plot a graph annotating the test results (optional).

Returns:

init_idx: AP threshold index in the spike_shape [dt]. a_plot: plot_abstract, if requested.

See also: `calcInitVm` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/24

A.41 Class `spike_shape_profile`

A.41.1 Constructor `spike_shape_profile/spike_shape_profile`

Summary: Holds the results profile from a spike_shape object.

Usage:

```
a_ss_profile = spike_shape_profile(results, a_spike_shape, props)
```

Parameters:

`results`**:** A structure containing test results.

`a_spike_shape`**:** A spike_shape object.

`props`**:** A structure with any optional properties.

Returns a structure object with the following fields:

results_profile: Contains results of tests. spike_shape: The spike_shape object from which results were obtained. props.

See also: `results_profile` (p. 182)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/08/17

A.41.2 Method `spike_shape_profile/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.41.3 Method `spike_shape_profile/plot_abstract`

Summary: Plots the spike shape with measurements marked in red.

Usage:

```
a_plot = plot_abstract(s, props)
```

Parameters:

`s`**:** A spike_shape object.

`props`**:** A structure with any optional properties.

- `absolute_peak_time`**:** Shift the peak to this point on the plot.
- `no_plot_spike`**:** Do not plot the spike shape first.

Returns:

a_plot: A plot_abstract object that can be visualized.

See also: `spike_shape` (p. 189), `plot_abstract` (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/08/17

A.42 Class `spikes`

A.42.1 Constructor `spikes/spikes`

Summary: Spike times from a trace.

Usage:

```
obj = spikes(times, num_samples, dt, id)
```

Parameters:

`times`**:** The spike times [dt].

`num_samples`**:** Number of samples in the original trace.

`dt`**:** Time resolution [s].

`id`**:** Identification string.

Returns a structure object with the following fields:

times, num_samples, dt, id.

See also: `trace/spikes` (p. 286), `trace` (p. 278), `spike_shape` (p. 189), `period` (p. 140)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/07/30

A.42.2 Method `spikes/plotData`

Summary: Plots a spikes object.

Usage:

```
a_plot = plotData(s, title_str)
```

Description: If s is a vector of spikes objects, returns a vector of plot objects.

Parameters:

`s`**:** A spikes object.

Returns:

a_plot: A plot_abstract object that can be visualized. title_str: (Optional) String to append to plot title.

See also: `trace` (p. 278), `plot_abstract` (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/10/21

A.42.3 Method spikes/get

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.42.4 Method spikes/plot

Summary: Plots spikes.

Usage:

```
h = plot(t)
```

Parameters:

- `t`: A spikes object.
- `title_str`: (Optional) String to append to plot title.

Returns:

h: Handle to figure object.

See also: spikes (p. 202), plot_abstract (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.42.5 Method spikes/ISICV

Summary: Calculates the coefficient of variation (CV) of the inter-spike-intervals (ISI).

Usage:

```
cv = ISICV(s, a_period)
```

Parameters:

- `s`: A spikes object.
- `a_period`: The period where spikes were found (optional)

Returns:

cv: Coefficient of variation.

See also: spikes (p. 202), period (p. 140)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/13

A.42.6 Method spikes/getResults

Summary: Runs all tests defined by this class and return them in a structure.

Usage:

```
results = getResults(s)
```

Parameters:

s: A spikes object.

Returns:

results: A structure associating test names to values in ms and mV (or mA).

See also: spikes (p. 202)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/13

A.42.7 Method spikes/plotISIs

Summary: Plots a spikes object.

Usage:

```
a_plot = plotISIs(s, title_str)
```

Description: If s is a vector of spikes objects, returns a vector of plot objects.

Parameters:

s: A spikes object.

Returns:

a_plot: A plot_abstract object that can be visualized. title_str: (Optional) String to append to plot title.

See also: trace (p. 278), plot_abstract (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/10/21

A.42.8 Method spikes/set

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.42.9 Method `spikes/periodWhole`

Summary: Returns the boundaries of the whole period of spikes, s.

Usage:

```
whole_period = periodWhole(s)
```

Parameters:

`s`**:** A spikes object.

See also: `period` (p. 140), `spikes` (p. 202)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/07/30

A.42.10 Method `spikes/getISIs`

Summary: Calculates the firing rate of the spikes found in the given period with an averaged inter-spike-interval approach.

Usage:

```
isi = getISIs(s, period)
```

Parameters:

`s`**:** A spikes object.

`period`**:** The period where spikes were found (optional)

Returns:

isi: Inter-spike-interval vector [dt]

See also: `trace` (p. 278), `spikes` (p. 202), `period` (p. 140)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/03/09

A.42.11 Method `spikes/spikeRateISI`

Summary: Calculates the firing rate of the spikes found in the given period with an averaged inter-spike-interval approach.

Usage:

```
freq = spikeRateISI(s, trace_index, times, period)
```

Parameters:

`s`**:** A spikes object.

`period`**:** The period where spikes were found (optional)

Returns:

freq: Firing rate [Hz]

See also: `trace` (p. 278), `spikes` (p. 202), `period` (p. 140)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/03/09

A.42.12 Method `spikes/spikeRate`

Summary: Calculates the average firing rate [Hz] of the given spike train.

Usage:

```
freq = spikeRate(s, a_period)
```

Parameters:

`s`: A spikes object.

`a_period`: The period where spikes were found (optional)

Returns:

freq: Firing rate [Hz]

See also: `spikes` (p. 202), `period` (p. 140)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/03/09

A.42.13 Method `spikes/subsref`

Summary: Defines generic indexing for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.42.14 Method `spikes/withinPeriodWOffset`

Summary: Returns a spikes object valid only within the given period, keeps the offset.

Usage:

```
obj = withinPeriodWOffset(s, a_period)
```

Parameters:

`s`: A spikes object.

`a_period`: The desired period

Returns:

obj: A spikes object

See also: `spikes` (p. 202), `period` (p. 140)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/05/09

A.42.15 Method `spikes/spikeAmpSlope`

Summary: Calculates the time constant and steady-state value of the spike amplitude for slow inactivating decays.

Usage:

```
[a_tau, da_inf] = spikeAmpSlope(a_spikes, a_trace, a_period)
```

Parameters:

`a_spikes`**:** A spikes object.

`a_trace`**:** A trace object.

`a_period`**:** The desired period (optional)

Returns:

a_tau: Approximate amplitude decay constant. da_inf: Delta change in final spike peak value from initial.

See also: `period` (p. 140), `spikes` (p. 202), `trace` (p. 278)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/15

A.42.16 Method `spikes/display`

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.42.17 Method `spikes/vertcat`

Summary: Vertical concatanation [a_spikes;with_spikes;...] operator.

Usage:

```
a_spikes = vertcat(a_spikes, with_spikes, ...)
```

Description: Concatanates spike times of with_spikes with that of a_spikes. Overrides the built-in vertcat function that is called when [a_spikes;with_spikes] is executed.

Parameters:

a_spikes, with_spikes, ...: Spikes objects.

Returns:

a_spikes: A tests_spikes that contains times of all given spikes objects.

See also: `vertcat` (p. **??**), `spikes` (p. 202)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/08/16

A.42.18 Method `spikes/withinPeriod`

Summary: Returns a spikes object valid only within the given period, subtracts the offset.

Usage:

```
obj = withinPeriod(s, a_period)
```

Parameters:

`s`: A spikes object.

`a_period`: The desired period

Returns:

obj: A spikes object

See also: `spikes` (p. 202), `period` (p. 140)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/07/31

A.42.19 Method `spikes/intoPeriod`

Summary: Shifts the spikes times to be within the given period.

Usage:

```
obj = intoPeriod(s, a_period)
```

Description: Assuming this spikes object's length fits into the given period, it shifts all times to start from the beginning of the given period. This may be used to reconstruct the original spikes object from subperiods that were cut out previously, using the withinPeriod method.

Parameters:

`s`: A spikes object.

`a_period`: The desired period

Returns:

obj: A spikes object

See also: spikes (p. 202), period (p. 140)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/07/31

A.42.20 Method spikes/addSpikes

Usage:

```
s = addSpike(s, times)
```

Parameters:

s: A spikes object.

times: Times of spikes to add

Returns:

s: The updated object.

See also: spikes (p. 202)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/08/16

A.42.21 Method spikes/plotFreqVsTime

Summary: Plots a frequency-time graph from the spikes object.

Usage:

```
a_plot = plotFreqVsTime(s, title_str, props)
```

Description: If s is a vector of spikes objects, returns a vector of plot objects.

Parameters:

s: A spikes object.

title_str: (Optional) String to append to plot title.

props: A structure with any optional properties.

timeScale: 's' for seconds, or 'ms' for milliseconds.

type: If 'simple' plots 1/is for each spike time,
'manhattan' uses flat lines of 1/isi height between spike times (default). (others passed to plot_abstract)

Returns:

a_plot: A plot_abstract object that can be visualized. title_str: (Optional) String to append to plot title.

See also: `trace` (p. 278), `plot_abstract` (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/05/05

A.42.22 Method `spikes/SFA`

Summary: Calculates the spike frequency accommodation (SFA) of the inter-spike-intervals (ISI).

Usage:

```
sfa = SFA(s, a_period)
```

Description: SFA is the ration of the last ISI to the first ISI in the period.

Parameters:

`s`**:** A spikes object.

`a_period`**:** The period where spikes were found (optional)

Returns:

sfa: Spike frequency accommodation.

See also: `spikes` (p. 202), `period` (p. 140)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/13

A.43 Class `spikes_db`

A.43.1 Constructor `spikes_db/spikes_db`

Summary: A database of spike shape results obtained from a period in a trace.

Usage:

```
a_spikes_db = spikes_db(data, col_names, a_trace, a_period, id, props)
```

Description: This is a subclass of tests_db. Use trace/analyzeSpikesInPeriod to get an instance of this class.

Parameters:

`data`**:** Database contents.

`col_names`**:** The column names.

`a_trace`: The trace where the spikes were found.

`a_period`: The period inside a_trace where spikes were found.

`id`: An identifying string.

`props`: A structure with any optional properties.

Returns a structure object with the following fields:

tests_db, trace, period, props.

See also: `tests_db` (p. 227), `trace` (p. 278), `period` (p. 140), `trace/analyzeSpikesInPeriod` (p. 282)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/08/17

A.43.2 Method `spikes_db/plot_abstract`

Summary: Visualizes the spikes_db by marking spike shapes measurements on the trace plot.

Usage:

```
a_pm = plot_abstract(a_db, title_str, props)
```

Parameters:

`a_db`: A spikes_db object.

`title_str`: (Optional) A string to be concatanated to the title.

`props`: A structure with any optional properties passed to trace/plotData.

Returns:

a_pm: A trace plot.

See also: `plot_abstract/plot_abstract` (p. 157), `tests_db/plot_abstract` (p. 276), `plotFigure` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/08/17

A.44 Class `sql_portal`

A.44.1 Constructor `sql_portal/sql_portal`

Summary: For import and export to external SQL engines using the Matlab Database Toolbox.

Usage:

```
obj = sql_portal(db_conn, id, props)
```

Description: Uses the Database (DB) Toolbox's connection object to read and write to external SQL engines.

Parameters:

`db_conn`**:** An object of the database class of the DB Toolbox.

`id`**:** An identifying string.

`props`**:** A structure with any optional properties.

Returns a structure object with the following fields:

db_conn: The DB Toolbox connection object. id, props.

See also: `tests_db` (p. 227), `database` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/11/29

A.44.2 Method `sql_portal/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.44.3 Method `sql_portal/sql_table`

Summary: Create an SQL table from the contents of a tests_db object.

Usage:

```
sql_table(a_sql_portal, a_tests_db, table_label, props)
```

Description: Converter function to get a tests_db object properly annotated with the metadata obtained from the results of the executed SQL query. Currently this function is limited to importing numeric data only.

Parameters:

`a_sql_portal`**:** A sql_portal object.

`a_tests_db`**:** A tests_db object.

`table_label`**:** Name of the newly created table.

`props`**:** A structure with any optional properties.

Returns:

See also: `tests_db/tests_db` (p. 227), `database` (p. **??**), `sql_portal/tests_db` (p. 213)

Author: Cengiz Gunay <cgunay@emory.edu>, 2008/12/17

A.44.4 Method `sql_portal/subsref`

Summary: Defines generic indexing for objects.

A.44.5 Method `sql_portal/sql_statement`

Summary: Run an SQL statement and discard its results.

Usage:

```
response_str = sql_statement(a_sql_portal, statement_string, props)
```

Description: This function is for sending SQL statements that do not return any data, for such functions as inserting data into a database, creating views, and running administration commands. See sql_portal/tests_db to import select query results into Pandora.

Parameters:

`a_sql_portal`**:** A sql_portal object.

`statement_string`**:** An SQL statement that does not return data.

`props`**:** A structure with any optional properties.

Returns:

response_str: Response string from the SQL engine.

See also: `sql_portal/tests_db` (p. 213), `database` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2008/04/18

A.44.6 Method `sql_portal/tests_db`

Summary: Create a tests_db object from the results of a SQL query.

Usage:

```
a_db = tests_db(a_sql_portal, query_string, props)
```

Description: Converter function to get a tests_db object properly annotated with the metadata obtained from the results of the executed SQL query. Currently this function is limited to importing numeric data only.

Parameters:

`a_sql_portal`**:** A sql_portal object.

`query_string`**:** An SQL query returning numeric results.

`query_id`: Identifier associated witht the query, to be passed
into the tests_db object.

`props`: A structure with any optional properties passed to tests_db.

Returns:

a_db: A tests_db object.

See also: `tests_db` (p. 227), `database` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/11/29

A.45 Class `stats_db`

A.45.1 Constructor `stats_db/stats_db`

Summary: A database of rows corresponding to statistical distribution properties of tests. Multiple pages can be used to indicate another dimension.

Usage:

```
a_stats_db = stats_db(test_results, col_names, row_names, page_names, id, props)
```

Description: This is a subclass of tests_3D_db. Allows generating a plot, etc.

Parameters:

`test_results`: The 3-d array of rows, columns, and pages.

`col_names`: Test names in this db.

`row_names`: Statistical test names for each row.

`page_names`: Meaning of each separate page of data
(e.g., a different invariant parameter).

`id`: An identifying string.

`props`: A structure with any optional properties.

`axis_limits`: Limits in the form of [xmin xmax ymin ymax]
for errorbar axes.

`yTicksPos`: 'left' means only put y-axis ticks to leftmost plot.

`xTicksPos`: 'bottom' means only put x-axis ticks to lowest plot.

Returns a structure object with the following fields:

tests_3D_db.

See also: `tests_3D_db` (p. 220), `plot_abstract` (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/07

A.45.2 Method stats_db/get

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.45.3 Method stats_db/compareStats

Summary: Merges multiple stats_dbs into pages of a single stats_db for comparison.

Usage:

```
a_mult_stats_db = compareStats(a_stats_db, a_2nd_stats_db, ...)
```

Description: Generates a plot_simple object from this histogram.

Parameters:

a_stats_db: A stats_db object.

Returns:

a_mult_stats_db: A multi-page stats_db.

See also: plot_abstract (p. 157), plot_simple (p. 169)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.45.4 Method stats_db/plotVarMatrix

Summary: Create a stack of parameter-test variation plots organized in a matrix.

Usage:

```
a_plot_stack = plotVarMatrix(p_stats, props)
```

Description: Skips the 'ItemIndex' test.

Parameters:

p_stats: Array of invariant parameter databases obtained from
calling tests_3D_db/paramsTestsHistsStats.

props: A structure with any optional properties, passed to plot_stack.

plotMethod: 'plotVar' uses stats_db/plotVar (default)
'plot_bars' uses stats_db/plot_bars

rotateYLabel: Rotate row labels this much (default=60).

Returns:

a_plot_stack: A plot_stack with the plots organized in matrix form

See also: paramsTestsHistsStats (p. ??), params_tests_profile (p. 140), plotVar. (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/17

A.45.5 Method stats_db/plotColorVar

Summary: Create a color-plot of parameter-test variations in a matrix.

Usage:

```
a_plot = plotColorVar(p_stats, props)
```

Description: Skips the 'ItemIndex' test.

Parameters:

p_stats: Array of invariant parameter databases obtained from calling tests_3D_db/paramsTestsHistsStats.

title_str: (Optional) String to append to plot title.

props: A structure with any optional properties, passed to plot_stack.

plotMethod: 'plotVar' uses stats_db/plotVar (default)
'plot_bars' uses stats_db/plot_bars

Returns:

a_plot: A plot_abstract with the color plot

See also: paramsTestsHistsStats (p. ??), params_tests_profile (p. 140), plotVar. (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/17

A.45.6 Method stats_db/set

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.45.7 Method stats_db/onlyRowsTests

Summary: Returns a tests_db that only contains the desired tests and rows (and pages).

Usage:

```
obj = onlyRowsTests(obj, rows, tests, pages)
```

Description: Selects the given dimensions and returns in a new tests_db object.

Parameters:

obj: A tests_db object.

rows: A logical or index vector of rows. If ':', all rows.

tests: Cell array of test names or column indices. If ':', all tests.

pages: (Optional) A logical or index vector of pages. ':' for all pages.

Returns:

obj: The new tests_db object.

See also: subsref (p. **??**), tests_db (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/17

A.45.8 Method stats_db/subsref

Summary: Defines generic indexing for objects.

A.45.9 Method stats_db/plotVar

Summary: Generates a plot of the variation between two tests.

Usage:

```
a_plot = plotVar(a_stats_db, test1, test2, props)
```

Description: Creates a plot description where the mean values are used for solid lines and the std values of test2 is indicated with errorbars. It is assumed that each page of the stats_db contains a value to be matched.

Parameters:

a_stats_db: A stats_db object.

test1: Test column for the x-axis, only mean values are used.

test2: Test column for the y-axis, std values are indicated with errorbars.

title_str: (Optional) String to append to plot title.

props: Optional properties.

plotType: 1, only errorbars (default); 2, errorbars extending from bars.

quiet: If 1, only display given title_str.

(rest passed to plot_abstract)

Returns:

a_plot: A plot_abstract object or one of its subclasses.

See also: plotVar (p. **??**), plot_simple (p. 169)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/13

A.45.10 Method stats_db/plot_bars_ax

Summary: Bar plot with extending errorbars for all columns in the same axis.

Usage:

```
a_plot = plot_bars_ax(a_tests_db, row, props)
```

Description: Differs from stats_db/plot_bars because it does not open a new axis for each column. This is only suitable if all columns have similar extents.

Parameters:

a_stats_db: A stats_db object.

title_str: Optional title string.

props: A structure with any optional properties.

putLabels: Put special column name labels.

Returns:

a_plot: A plot_abstract object that can be plotted.

See also: plot_abstract (p. 157), plotFigure (p. **??**), stats_db/plot_bars (p. 219)

Author: Cengiz Gunay <cgunay@emory.edu>, 2008/04/17

A.45.11 Method stats_db/plotYTests

Summary: Create an errorbar plot of database stats measures against given X-axis values.

Usage:

```
a_p = plotYTests(a_stats_db, x_vals, tests, axis_labels, title_str, short_title, command,
props)
```

Parameters:

`a_stats_db`: A params_tests_db object.

`x_vals`: A vector of X-axis values.

`tests`: A vector or cell array of columns to correspond to each value from x_vals.

`title_str`: (Optional) A string to be concatanated to the title.

`short_title`: (Optional) Few words that may appear in legends of multiplot.

`command`: (Optional) Command to do the plotting with (default: 'plot')

`props`: A structure with any optional properties.

  `LineStyle`: Plot line style to use. (default: 'd-')

  `quiet`: If 1, don't include database name on title.

Returns:

a_p: A plot_abstract.

Example:

```
» a_p = plotYTests(a_stats_db, [0 40 100 200], ...
'IniSpontSpikeRateISI_0pA', 'PulseIni100msSpikeRateISI_D40pA', ...
'PulseIni100msSpikeRateISI_D100pA', 'PulseIni100msSpikeRateISI_D200pA', ...
'current pulse [pA]', 'firing rate [Hz]', ', f-I curves', 'neuron 1');
» plotFigure(a_p);
```

See also: plotFigure (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/01/23

A.45.12 Method stats_db/plot_bars

Summary: Creates a bar graph with errorbars for each db column.

Usage:

```
a_plot = plot_bars(a_stats_db, title_str, props)
```

Description: Looks for 'min', 'max', and 'STD' labels in the row_idx for drawing the errorbars. Each page of the DB will produce grouped bars.

Parameters:

`a_stats_db`: A stats_db object.

`title_str`: The plot title.

`props`: A structure with any optional properties.

  `pageVariable`: The column used for denoting page values. (passed to plot_bars/plot_bars)

Returns:

a_plot: A object of plot_bars or one of its subclasses.

See also: plot_abstract (p. 157), plot_bars/plot_bars (p. 163)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.45.13 Method stats_db/plot_abstract

Summary: Generates an error bar graph for each db columns.

Usage:

```
a_plot = plot_abstract(a_stats_db, title_str, props)
```

Description: Generates a plot_simple object from this histogram. Looks for 'mean', 'min', 'max', and 'STD' labels in the row_idx for drawing the errorbars. Each column of a_stats_db is shown in a separate axis. Values from multiple pages of a_stats_db are shown as distinct points in the axis.

Parameters:

a_stats_db: A histogram_db object.

title_str: A title string on the plot

props: A structure with any optional properties.

Returns:

a_plot: A object of plot_abstract or one of its subclasses.

See also: plot_abstract (p. 157), plot_simple (p. 169)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.46 Class tests_3D_db

A.46.1 Constructor tests_3D_db/tests_3D_db

Summary: A database multiple pages with rows of test columns. Each page may represent aspects of the data that are different, but not defined in this object.

Usage:

```
a_3D_db = tests_3D_db(data, col_names, row_names, page_names, id, props)
```

Description: This is a subclass of tests_db. Usually it contains a RowIndex column that points to an original db from which this data originated. The row indices can be used to reach the values associated with different pages of information contained in this object.

Parameters:

- `data`**:** The 3-d vector of rows, columns, and pages.
- `col_names`**:** Colun names of the database.
- `id`**:** An identifying string.
- `props`**:** A structure with any optional properties.
  - `invarName`**:** Name of the invariant parameter for this db.

Returns a structure object with the following fields:

tests_db, page_idx.

See also: `tests_db` (p. 227), `tests_db/invarValues` (p. 231)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/30

A.46.2 Method `tests_3D_db/get`

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.46.3 Method `tests_3D_db/plotParamPairImage`

Summary: Generates an image plot of variation of a test with two parameters in the first page.

Usage:

```
a_plot = plotParamPairImage(a_db, test, title_str, props)
```

Description: It is assumed that the 3D DB is created by invariant combinations of two parameters, which are the first two columns. Each page of the db must contain a same parameter values. This is the default character of tests_3D_db created by params_tests_db/invarParam. Parameter values will be enumerated and then an image plot is created.

Parameters:

- `a_db`**:** A tests_3D_db object.
- `test`**:** Test column to take the measure value.
- `title_str`**:** (Optional) String to append to plot title.
- `props`**:** Optional properties to be passed to plot_abstract.
  - `truncateDecDigits`**:** Truncate labels to this many decimal digits.
  - `labelSteps`**:** Skip this many labels between ticks to reduce to total number.

`maxValue`**:** Maximal value to normalize colors and to annotate the colorbar.

Returns:

a_plot: A plot_abstract object or one of its subclasses.

Example:

```
Find relationship of two parameters against a measure:
» plotFigure(plotParamPairImage(invarParam(a_db, 'NaF', 'KCNQ'),
'PulseIni100msRest2SpikeRateISI_D100pA'));
```

See also: `params_tests_db/invarParam` (p. 131), `plotImage` (p. **??**), `plot_abstract.` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/10

A.46.4 Method `tests_3D_db/mergePages`

Summary: Merges tests from separate pages into a 2D params_tests_db.

Usage:

```
a_db = mergePages(db, page_tests, page_suffixes)
```

Description: Keeps uniqueness by adding suffixes to test names. If you're using invarParams, do swapRowsPages, then join with original db to get the parameter values.

Parameters:

`db`**:** A tests_3D_db object.

`page_tests`**:** Cell array of list of tests to take from each page.

`page_suffixes`**:** Cell array of suffixes to append to tests from each page.

Returns:

a_db: A tests_db object.

See also: `tests_db` (p. 227), `tests_3D_db` (p. 220)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/01/13

A.46.5 Method `tests_3D_db/diff2D`

Summary: Creates a tests_db by taking the derivative of the given test.

Usage:

```
a_tests_db = diff2D(a_db, test, props)
```

Description: Applies the diff function to the chosen test, and collapses the middle dimension of the 3D DB to create a 2D DB and transposes it. The result is that the pages of the 3D DB becomes the rows of the new database, and the differenced rows appear as new columns, each named uniquely. The column index would correspons to the row index in the 3D DB. A new column 'PageNumber' is appended to point back to the 3D DB.

Parameters:

`a_db`**:** A tests_3D_db object.

`test`**:** Test column.

`props`**:** Optional properties.

Returns:

a_tests_db: A tests_db that holds the requested differences of parameter values.

See also: `boxplot` (p. **??**), `plot_abstract` (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/05/22

A.46.6 Method `tests_3D_db/set`

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.46.7 Method `tests_3D_db/histograms`

Usage:

```
a_histogram_db = histogram(db, col, num_bins)
```

Description: If one wants to get histograms of test values for each single value of the selected invariant parameter, then swapRowsPages should be done first on db.

Parameters:

`db`**:** A tests_3D_db object.

`col`**:** Column to find the histogram.

`num_bins`**:** Number of histogram bins (Optional, default=100)

Returns:

a_histogram_db: A histogram_db object containing the histogram.

See also: `histogram_db` (p. 96), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/04

A.46.8 Method `tests_3D_db/plotVarBox`

Summary: Generates a boxplot of the variation between two tests.

Usage:

```
a_plot = plotVarBox(a_db, test1, test2, notch, sym, vert, whis, props)
```

Description: It is assumed that each page of the db contains a different parameter value.

Parameters:

`a_db`**:** A tests_3D_db object.

`test1`**:** Test column for the x-axis, only mean values are used.

`test2`**:** Test column for the y-axis, used for boxplot.

`notch, sym, vert, whis`**:** See boxplot, defaults = (1, '+', 1, 1.5).

`props`**:** Optional properties to be passed to plot_abstract.

Returns:

a_plot: A plot_abstract object or one of its subclasses.

See also: `boxplot` (p. **??**), `plot_abstract` (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/10

A.46.9 Method `tests_3D_db/display`

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.46.10 Method `tests_3D_db/joinPages`

Summary: Joins the rows of the given db to the with_db rows matching with the PageIndex column.

Usage:

```
a_db = joinPages(db, tests, with_db, w_tests)
```

Description: Replicates the desired columns in the with_db with rows having a page index and joins them next to desired columns from the current 3D_db. Flattens the resulting 3D_db to become a 2D db. Assumes each page index only appears once in with_db.

Parameters:

`db`: A tests_3D_db object.

`with_db`: A tests_db object with a PageIndex column.

Returns:

a_db: A tests_db object.

See also: `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/15

A.46.11 Method `tests_3D_db/corrCoefs`

Summary: Calculates correlation coefficients by comparing col1 with other cols.

Usage:

```
a_coefs_db = corrCoefs(db, col1, cols, props)
```

Description: If db has multiple pages, then each page in db produces a row of coefficients and matching PageIndex. Assuming the db was created with invarValues, this function finds the invariant correlation coefficients between its columns. The invariant correlation coefficients are the correlation of one column value with another column value when some other column values are fixed. Since there are many occurences of the invariant coefficients, a histogram can then be created and returned from the created db. The other columns that are fixed are not in this db object, but can be reached using the indices to the original db. The page number is saved in the created db, so that it can be used to find the page from which the coefficient came. Then row indices of the page points to original constant column values.

Parameters:

`db`: A tests_db object.

`col1`: Column to compare.

`cols`: Columns to be compared with col1.

`props`: A structure with any optional properties.

`skipCoefs`: If 1, coefficients of less confidence than will be skipped. (default=1)

Returns:

a_coefs_db: A corrcoefs_db of the coefficients and page indices.

See also: `tests_db` (p. 227), `corrcoefs_db` (p. 82)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/30

A.46.12 Method `tests_3D_db/paramsTestsHistsStats`

Summary: Calculates histograms and statistics for DB.

Usage:

```
[pt_hists, p_stats] = paramsTestsHistsStats(p_t3ds, props)
```

Description: Calculates histograms and statistics for all combinations of tests and params and returns them in a cell array. Skips the 'ItemIndex' test.

Parameters:

`p_t3ds`: Array of invariant parameter databases obtained by calling the params_tests_db/invarParams method.

`props`: Optional properties.

`statsMethod`: method to call to get a stats_db (default='statsMeanSE')

`useDiff`: If 1, takes the derivative with diff on the 3D DBs (default=0).

Returns:

pt_hists: An array of 3D histograms for each pair of param and test. p_stats: An array of stats_dbs for each param.

See also: `invarParams` (p. **??**), `params_tests_profile` (p. 140)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/17

A.46.13 Method `tests_3D_db/swapRowsPages`

Summary: Swaps the row dimension with the page dimension of the tests_3D_db.

Usage:

```
a_3D_db = swapRowsPages(db)
```

Description: Assuming that this is a invariant parameter and tests relations db, this function transposes the data matrix by swapping the pages with rows. Each resulting page correspond to a single value of the chosen parameter, with each row containing a test result with different combinations of the rest of the parameters.

Parameters:

`db`**:** A tests_db object.

Returns:

a_3D_db: A tests_3D_db object.

See also: `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/04

A.47 Class `tests_db`

A.47.1 Constructor `tests_db/tests_db`

Summary: A generic database of test results organized in a matrix format.

Usage:

```
obj = tests_db(test_results, col_names, row_names, id, props)
```

Description: Defines all operations on this structure so that subclasses can use them.

Parameters:

`test_results`**:** A matrix that contains columns associated with tests and rows for separate observations.

`col_names`**:** Cell array of column names of test_results.

`row_names`**:** Cell array of row names of test_results.

`id`**:** An identifying string.

`props`**:** A structure with any optional properties.

Returns a structure object with the following fields:

data: The data matrix. row_idx, col_idx: Structure associating row/column names to indices. id, props.

See also: params_tests_db (p. 118), params_db (p. **??**), test_variable_db (N/I) (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/01

A.47.2 Method tests_db/setRows

Summary: Sets the rows of observations in tests_db.

Usage:

```
index = setRows(obj, rows)
```

Description: Sets a new set of observations to the database and returns the new DB.

Parameters:

obj: A tests_db object.

rows: A matrix that contains rows for the DB.

Returns:

obj: The tests_db object with the new rows.

See also: allocateRows (p. **??**), addRow (p. **??**), tests_db (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/08

A.47.3 Method tests_db/corrcoef

Summary: Calculates a correlation coefficient matrix by comparing cols.

Usage:

```
a_coefs_db = corrcoef(db, cols, props)
```

Parameters:

db: A tests_db object.

cols: Columns to be compared.

props: A structure with any optional properties.

skipCoefs: If 1, coefficients of less confidence than will be skipped. (default=1)

alpha: Skip coefs with p values lower than this (default=0.05).

Returns:

a_coefs_db: A tests_3D_db of the coefficients.

See also: tests_db (p. 227), corrcoefs_db (p. 82)

Author: Cengiz Gunay <cgunay@emory.edu>, 2008/04/25

A.47.4 Method `tests_db/allocateRows`

Summary: Preallocates a NaN-filled num_rows rows in tests_db.

Usage:

```
obj = allocateRows(obj, num_rows)
```

Description: Allocates the desired number of rows to speed up filling up the data matrix using assignRowsTests. Using addRow after this operation is still expensive. The method of allocating a matrix, filling it up, and then providing it to the tests_db constructor is the preferred method of creating tests_db objects.

Parameters:

`obj`: A tests_db object.

`num_rows`: The predicted number of observations for this tests_db.

Returns:

obj: The new tests_db object.

See also: `assignRowsTests` (p. **??**), `addRow` (p. **??**), `setRows` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/08

A.47.5 Method `tests_db/matchingRow`

Summary: Creates a criterion database for matching the tests of a row.

Usage:

```
crit_db = matchingRow(db, row, props)
```

Description: Copies selected test values from row as the first row into the criterion db. Adds a second row for the STD of each column in the db. Calculates the covariance for using the Mahalonobis distance in the ranking.

Parameters:

`db`: A tests_db object.

`row`: A row index to match.

`props`: A structure with any optional properties.

  `distDB`: Take the standard deviation and covariance of this db instead.

Returns:

crit_db: A tests_db with two rows for values and STDs.

Example:

```
» crit_db = matchingRow(phys_control_compare_db,
find(phys_control_compare_db(:, 'TracesetIndex') == 61))
```

See also: rankMatching (p. ??), tests_db (p. 227), tests2cols (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/12/08

A.47.6 Method tests_db/get

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.47.7 Method tests_db/tests2idx

Summary: Find dimension indices from a test names/numbers specification.

Usage:

```
idx = tests2idx(db, dim_name, tests)
```

Parameters:

db: A tests_db object.

dim_name: String indicating 'row', 'col', or 'page'

tests: Either a single or array of column numbers, or a single
test name or a cell array of test names. If ':', all tests. For name strings, regular expressions are supported if quoted with slashes (e.g., '/a.*/')

Returns:

idx: Array of column indices.

Example:

```
» cols = tests2idx(a_db, 'col', 'col1', '/col2+/');
will return indices of col1 and columns like col2, col22, col22, etc.
```

See also: tests_db (p. 227), tests2cols (p. ??), regexp (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/07

A.47.8 Method `tests_db/invarValues`

Summary: Finds all sets in which given columns vary while the rest are invariant.

Usage:

```
a_tests_3D_db = invarValues(db, cols, in_page_unique_cols)
```

Description: Useful when trying to find relationships between some columns independent of other columns. In a database that contains results of a multivariate function, this function can find the effect of one or more parameters when other parameters are kept constant (i.e., invariant). Rows with the values of the desired columns are separated into the pages of a tests_3D_db for each unique set of the other column values. These invariant values of the other columns are missing from the resulting tests_3D_db, instead a RowIndex is kept pointing to the db in which they can be found. See joinRows for joining the results back with the invariant columns. In databases that contain all unique combinations of certain parameters, the resulting 3D database becomes symmetric. This function row-sorts the database to ensure that each page has the same parameter values in the same rows. This is important because when the rows and pages of database is swapped (see tests_3D_db/swapRowsPages) each page has the same value of the in_page_unique_cols variables. Other functions such as tests_3D_db/mergePages also depend on this property. However, for databases with missing combinations, in_page_unique_cols specifies which columns is used to guide which rows of the page to place values found. This function will fail if you do not have such a column. Note: the trial column will be ignored before finding invariant values.

Parameters:

`db`: A tests_db object.

`cols`: Vector of column numbers to find values when others are invariant. Include result columns here.

`in_page_unique_cols`: Vector of columns that have the same unique values in each page

(Optional; used only if database is not symmetric, to ignore missing values of in_page_unique_cols)

Returns:

a_tests_3D_db: A tests_3D_db object of organized values.

Example:

```
» a_db = tests_db([ ... ], 'par1', 'par2', 'measure1', 'measure2')
» a_3d_db = invarValues(a_db, [2:4], 'par2')
» » joined_3d_db = joinRows(a_db, a_3d_db)
» displayRows(joined_3d_db(:, :, 1))
```

See also: tests_3D_db (p. 220), tests_3D_db/corrCoefs (p. 225), tests_3D_db/plotPair (p. ??), joinRows (p. ??), tests_3D_db/swapRowsPages (p. 227), tests_3D_db/mergePages (p. 222)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/30

A.47.9 Method `tests_db/shufflerows`

Summary: Returns a db with rows of given test columns are shuffled.

Usage:

```
s = shufflerows(db, tests, grouped)
```

Description: Can be used for shuffle prediction. Basically, shuffle rows of tests and rerun high order analyses.

Parameters:

`db`**:** A tests_db object.

`tests`**:** Tests to shuffle.

`grouped`**:** If 1 then shuffle tests all together,
if 0 shuffle each test separately.

Returns:

a_db: The shuffled db.

See also: tests_db (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/10

A.47.10 Method `tests_db/plot`

Summary: Generic method to plot a tests_db or a subclass. Requires a plot_abstract method to be defined for this object.

Usage:

```
h = plot(a_tests_db, title_str, props)
```

Parameters:

`a_tests_db`**:** A histogram_db object.

`title_str`**:** (Optional) String to append to plot title.

`props`**:** A structure with any optional properties, passed to plot_abstract.

Returns:

h: The figure handle created.

See also: `plot_abstract` (p. 157), `plotFigure` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/06

A.47.11 Method `tests_db/setProp`

Summary: Generic method for setting optional object properties.

Usage:

```
obj = setProp(obj, prop1, val1, prop2, val2, ...)
```

Description: Modifies or adds property values. As many property name-value pairs can be specified.

Parameters:

`obj`**:** Any object that has a props field.

`attr`**:** Property name

`val`**:** Property value.

Returns:

obj: The new object with the updated properties.

See also:

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/22

A.47.12 Method `tests_db/physiol_bundle`

Summary: Create a physiol_bundle from a raw physiology database.

Usage:

```
a_pbundle = physiol_bundle(phys_dball, phys_dataset, props)
```

Description: Removes small bias currents, calculates input resistance by averaging negative CIP traces, averages multiple traces with similar treatments, selects certain CIP levels collapses its rows to create a one-neuron-per-pow database. It includes post-DB calculated columns such as rate ratios between spont and recovery periods.

Parameters:

`phys_dball`**:** A raw database obtained by loading traces from the tracesets.

`phys_dataset`**:** Dataset object passed to physiol_bundle.

`props`**:** Optional parameters.

`weedCols`**:** Cell array of parameter columns to be weed-out before averaging rows
that are same w.r.t other parameters. (default=’pulseOn’, ’pulseOff’, ’traceEnd’, ’pAbias’, ’ItemIndex’).

`drugCols`**:** Cell array of drug names that need to be zero for the
control db (default=’TTX’, ’Apamin’, ’EBIO’, ’XE991’, ’Cadmium’, ’drug_4AP’).

`CIPList`**:** row array specifying the CIP levels to choose (eliminate the

others), default is an empty array, which means to choose all.

`biasLimit`**:** Limit in pA, biases larger +/- than which will be
eliminated. (default=30)

Returns:

phys_joined_db: Final one row per cip and neuron db. phys_joined_control_db: Rows where all drug treatments are zero. phys_db: Original db only with parameter and including the weedCols.

See also: `physiol_bundle` (p. 142), `params_tests_db` (p. 118)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/12/21

A.47.13 Method `tests_db/mtimes`

Summary: Multiplies the DB with a scalar.

Usage:

```
a_db = mtimes(left_obj, right_obj)
```

Parameters:

`left_obj, right_obj`**:** Operands of the multiplication. One or more must be of type tests_db.

Returns:

a_db: The resulting tests_db.

See also: `tests_db/times` (p. 250), `mtimes` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/05/24

A.47.14 Method tests_db/addColumns

Summary: Inserts new columns to tests_db.

Description: Adds new test columns to the database and returns the new DB. Usage 2 concatanates two DBs columnwise. This operation is expensive in the sense that the whole database matrix needs to be enlarged just to add a single new column. The method of allocating a matrix, filling it up, and then providing it to the tests_db constructor is the preferred method of creating tests_db objects. This method may be used for measures obtained by operating on raw measures.

Parameters:

`obj, b_obj`: A tests_db object.

`test_names`: A cell array of test names to be added.

`test_columns`: Data matrix of columns to be added.

Returns:

obj: The tests_db object that includes the new columns.

See also: `allocateRows` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/09/30

A.47.15 Method tests_db/rdivide

Summary: Adds a DB to another or to a scalar.

Usage:

```
a_db = rdivide(left_obj, right_obj)
```

Description: If DBs have mismatching columns only the common columns will be kept. In any case, the resulting DB columns will be sorted in the order of the left-hand-side DB.

Parameters:

`left_obj, right_obj`: Operands of the addition. One must be of type tests_db and the other can be a scalar or tests_db.

Returns:

a_db: The resulting tests_db.

See also: `rdivide` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/12/13

A.47.16 Method tests_db/tests2log

Summary: Return logical array of indices from a test names/numbers specification.

Usage:

```
a_log = tests2log(db, dim_name, tests)
```

Description: See tests2idx.

Parameters:

db: A tests_db object.

dim_name: String indicating 'row', 'col', or 'page'

tests: Either a single or array of column numbers, or a single
test name or a cell array of test names. If ':', all tests. For name strings, regular expressions are supported if quoted with slashes (e.g., '/a.*/')

Returns:

a_log: Array of column indices.

Example:

```
» cols = tests2log(a_db, 'col', 'col1', '/col2+/');
» stripped_db = a_db(:,  cols)
will remove columns col1 and col2, col22, col22, etc.  from stripped_db.
```

See also: tests_db (p. 227), tests2cols (p. ??), regexp (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2008/05/27

A.47.17 Method tests_db/transpose

Summary: Transposes data matrix and swaps row and columns metadata as well.

Usage:

```
a_db = transpose(a_db)
```

Parameters:

a_db: A tests_db.

Returns:

a_db: The transposed tests_db.

See also: transpose (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/02/07

A.47.18 Method `tests_db/joinRows`

Summary: Joins a_db rows with w_db rows having matching RowIndex values.

Usage:

```
a_db = joinRows(a_db, w_db, props)
```

Description: Concatenates columns of rows matching the join condition from the two databases. Each row index must appear only once in w_db. The created db preserves the ordering of w_db. See the multipleIndices option if there are several redundant index columns. Multiple pages in w_db are accepted (see keepNaNs option). This function is the equivalent of a "right outer join" command in SQL, w_db being the database table on the right.

Parameters:

`a_db`**:** A tests_db object.

`w_db`**:** A tests_db object with a row index column.

`props`**:** A structure with any optional properties.

- `indexColName`**:** (Optional) Name of row index column (default='RowIndex').
- `keepNaNs`**:** If 1, substitute NaN values for NaN indices. (default=1, for multi-page DBs; 0, otherwise).
- `multipleIndices`**:** If 1, search for substitute RowIndex* columns for indices with NaN values. It will fail if all indices are NaNs. (default=0)

Returns:

a_db: A tests_db object.

See also: `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/16

A.47.19 Method `tests_db/histogram`

Summary: Returns histogram of chosen database column.

Usage:

```
a_histogram_db = histogram(db, col, num_bins, props)
```

Description: Generates a histogram_db object with rows corresponding to histogram entries. If an array of DBs is given, finds and uses common histogram bin centers.

Parameters:

`db`**:** A tests_db object.

`col`**:** Column to find the histogram.

`num_bins`**:** Number of histogram bins (Optional, default=100), or vector of histogram bin centers.

`props`**:** A structure with any optional properties.

`normalized`**:** If 1, normalize histogram counts.

Returns:

a_histogram_db: A histogram_db object containing the histogram.

Example:

```
» a_hist_db = histogram(my_db, 'spike_width');
» plot(a_hist_db);
```

See also: `histogram_db` (p. 96), `tests_db` (p. 227), `hist` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/17

A.47.20 Method `tests_db/enumerateColumns`

Summary: Replaces each value with an integer pointing to the index of enumerated unique values in a column.

Usage:

```
a_db = enumerateColumns(a_db, tests, props)
```

Description: Finds unique values of each column, and replaces the original values with the enumerated indices of these unique values. Useful for normalizing all parameter values in a hypercube.

Parameters:

`a_db`**:** A tests_db object.

`tests`**:** Array of tests to be enumerated.

`props`**:** Optional properties.

`truncateDecDigits`**:** Use only up to this many decimal digits after the point
when checking for uniqueness.

Returns:

a_db: The modified DB.

Example:

```
» enumerated_db = enumerateColumns(a_db(:, 1:9));
```

See also: uniqueValues (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/06/14

A.47.21 Method `tests_db/uop`

Summary: Unary operation.

Usage:

```
a_db = uop(left_obj, op_func, op_id)
```

Description: Applies the operation to the database contents and updates its id field. Unary minus (uminus) uses this function.

Parameters:

`left_obj`: Operands of the operation.

`op_func`: Operation function (e.g., @plus).

`op_id`: A string to represent the operation that will show up in the returned id.

Returns:

a_db: The resulting tests_db.

See also: tests_db/uminus (p. 255), uminus (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2008/01/16

A.47.22 Method `tests_db/approxMappingNNet`

Summary: Approximates the desired input-output mapping using a Matlab neural network.

Usage:

```
[an_approx_db, a_nnet] = approxMappingNNet(a_db, input_cols, output_cols, props)
```

Description: Approximates the mapping between the given inputs to outputs using the Matlab Neural Network Toolbox. By default it creates a feed-forward network to be trained with a Levenberg-Marquardt training algorithm (see newff). Returns and the trained network object and a database with output columns obtained from the approximator. The outputs can then be compared to the original database to test the success of the approximation. If 'warning on verbose' is issued prior to running, it provides additional debug info.

Parameters:

`a_db`**:** A tests_db object.

`input_cols, output_cols`**:** Input and output columns to be mapped (see tests2cols for accept column specifications).

`props`**:** A structure with any optional properties.

`nnetFcn`**:** Neural network classifier function (default='newff')

`nnetParams`**:** Cell array of parameters passed to nnetFcn after inputs and outputs.

`trainMode`**:** 'batch' or 'incr'.

`classProbs`**:** 'prob': use probabilistic sampling to normalize prior class probabilities.

`maxEpochs`**:** maximum number of epochs to train for. (Rest passed to balanceInputProbs and tests_db)

Returns:

an_approx_db: A tests_db object containing the original inputs and the approximated outputs. a_nnet: The Matlab neural network approximator object.

Example:

```
» [a_class_db, a_nnet = approxMappingNNet(my_db, 'NaF', 'Kv3', 'spike_width');
» plotFigure(plot_superpose(plotScatter(my_db, 'NaF', 'spike_width'),
plotScatter(a_class_db, 'NaF', 'spike_width')))
```

See also: `tests_db` (p. 227), `newff` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/12/12

A.47.23 Method `tests_db/isnan`

Summary: Returns logical row indices of NaN-valued columns.

Usage:

```
rows = isnan(db, col)
```

Parameters:

`db`**:** A tests_db object.

`col`**:** Column to check (Optional, default = 1)

Returns:

rows: A logical column vector of rows.

See also: `isnan` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/06

A.47.24 Method tests_db/minus

Summary: Subtracts a DB from another or from a scalar.

Usage:

```
a_db = minus(left_obj, right_obj)
```

Description: If DBs have mismatching columns only the common columns will be kept. In any case, the resulting DB columns will be sorted in the order of the left-hand-side DB.

Parameters:

left_obj, right_obj: Operands of the subtraction. One must be of type tests_db
and the other can be a scalar.

Returns:

a_db: The resulting tests_db.

See also: minus (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/05/24

A.47.25 Method tests_db/displayRowsTeX

Summary: Generates a LaTeX table that lists rows of this DB.

Usage:

```
tex_string = displayRowsTeX(a_db, caption, props)
```

Description: By default table is rotated 90 degrees and scaled to 90

Parameters:

a_db: A tests_db object.

caption: Table caption.

props: A structure with any optional properties, passed to TeXtable.

Returns:

tex_string: LaTeX string for table float.

Example:

```
» string2File(displayRowsTeX(a_db(1:10, 4:7), 'some values',
struct('rotate', 0)), 'table.tex')
```

See also: displayRows (p. ??), TeXtable (p. ??), cell2TeX (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/12/16

A.47.26 Method `tests_db/plotXRows`

Summary: Create a scatter plot with a test versus the row numbers on the X-axis.

Usage:

```
a_p = plotXRows(a_db, test_y, title_str, short_title, props)
```

Parameters:

`a_db`**:** A params_tests_db object.

`test_y`**:** Y variable.

`title_str`**:** (Optional) A string to be concatanated to the title.

`short_title`**:** (Optional) Few words that may appear in legends of multiplot.

`props`**:** A structure with any optional properties passed to
plotScatter.

`RowName`**:** Label to show on X-axis, becomes a db column (default='RowNumber')

Returns:

a_p: A plot_abstract.

See also:

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/01/16

A.47.27 Method `tests_db/plotCovar`

Summary: Generates an image plot of the covariance-type values in a_db.

Usage:

```
a_plot = plotCovar(a_db, title_str, props)
```

Parameters:

`a_db`**:** A tests_db object that resulted from a function like cov.

`title_str`**:** (Optional) String to append to plot title.

`props`**:** Optional properties.

`inverse`**:** If 1, take inverse of the data matrix.

`corrcoef`**:** If 1, normalize matrix elements to get corrcoef values.

`logScale`**:** If 1, take logarithm of values before plotting.

`localityIters`**:** Apply a locality optimization algorithm with
this many iterations. (rest passed to plot_image.)

Returns:

a_plot: A plot_abstract object or one of its subclasses.

Example:

```
» plotFigure(plotCovar(cov(get(constrainedMeasuresPreset(pbundle2, 6),
'joined_control_db'))));
```

See also: `tests_db/cov` (p. 244), `plotImage` (p. **??**), `tests_db/matchingRow` (p. 229), `corrcoefs.` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/05/30

A.47.28 Method `tests_db/noNaNRows`

Summary: Returns a DB by removing rows containing any NaN or Inf.

Usage:

```
a_db = noNaNRows(a_db)
```

Parameters:

`a_db`**:** A tests_db object.

Returns:

a_db: DB with missing rows.

See also: `tests_db/isnanrows` (p. 262)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/09/21

A.47.29 Method `tests_db/princomp`

Summary: Generates a database of the principal components of given DB.

Usage:

```
a_pca_db = princomp(db, props)
```

Parameters:

`db`**:** A tests_db object.

`props`**:** A structure with any optional properties.

`normalized`**:** If specified zscore is used before princomp.

Returns:

a_pca_db: A tests_db where each row is a principal component.

See also: `princomp` (p. **??**), `zscore` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/09/21

A.47.30 Method `tests_db/cov`

Summary: Generates a database of the covariance of given DB.

Usage:

```
a_cov_db = cov(db, props)
```

Parameters:

`db`**:** A tests_db object.

`props`**:** A structure with any optional properties.

`keepOrigDB`**:** Keep db as origDB in the props.
(others passed to tests_db)

Returns:

a_cov_db: A tests_db which contains the covariance matrix.

See also: `cov` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/05/25

A.47.31 Method `tests_db/set`

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.47.32 Method `tests_db/plotrow`

Summary: Creates a plot_abstract describing the desired db row.

Usage:

```
a_plot = plotrow(a_tests_db, row, title_str, props)
```

Parameters:

`a_tests_db`**:** A tests_db object.

`row`**:** Row number to visualize.

`title_str`**:** Optional title string.

`props`**:** A structure with any optional properties.

`putLabels`**:** Put special column name labels.

Returns:

a_plot: A plot_abstract object that can be plotted.

See also: `plot_abstract` (p. 157), `plotFigure` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/08

A.47.33 Method `tests_db/checkConsistentCols`

Summary: Check if two DBs have exactly the same columns.

Usage:

```
[col_names, with_col_names] = checkConsistentCols(db, with_db)
```

Parameters:

`db`**:** A tests_db object.

`with_db`**:** A tests_db object whose column names are checked for consistency.

`props`**:** A structure with any optional properties.

`useCommon`**:** Tolerate mismatching column names and only return the common columns.

Returns:

col_names, with_col_names: list of column names of each DB.

See also: `vertcat` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/01/18

A.47.34 Method `tests_db/ne`

Summary: Returns logical indices of db rows that doesn't match with given row.

Usage:

```
rows = ne(db, row)
```

Parameters:

`db`**:** A tests_db object.

`row`**:** Row array to be compared with db rows.

Returns:

rows: A logical or index vector to be used in indexing db objects.

See also: `ne` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/17

A.47.35 Method `tests_db/std`

Summary: Returns the std of the data matrix of a_db. Ignores NaN values.

Usage:

```
[a_db, n] = std(a_db, sflag, dim)
```

Description: Does a recursive operation over dimensions in order to remove NaN values. This takes considerable amount of time compared with a straightforward std operation.

Parameters:

`a_db`**:** A tests_db object.

`dim`**:** Work down dimension.

Returns:

a_db: The DB with std values. n: (Optional) Numbers of non-NaN rows included in calculating each column.

See also: `std` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/06

A.47.36 Method `tests_db/sortrows`

Summary: Returns a sorted_db according to given columns.

Usage:

```
[sorted_db, idx] = sortrows(db, cols)
```

Description: WARNING: For multi-page dbs, sorts only the first page and applies the ordering to all other pages which may produce wrong results for some applications.

Parameters:

`db`**:** A tests_db object.

`cols`**:** Columns to use for sorting.

Returns:

sorted_db: The sorted tests_db. idx: The row index permutation vector, such that sorted_db = db(idx, :).

See also: `sortrows` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/11

A.47.37 Method `tests_db/processDimNonNaNInf`

Summary: Recursively process the specified dimension with the desired function after removing NaNs and Infs.

Usage:

```
[a_db, n] = processDimNonNaNInf(a_db, dim, a_func, a_func_name)
```

Description: Does a recursive operation over other dimensions in order to remove NaN and Inf values. This takes more time than applying the function directly.

Parameters:

`a_db`**:** A tests_db object.

`dim`**:** Work down dimension (see mean).

`a_func`**:** A function name or handle to be passed to feval that
takes the data as the first argument and dimension to work as second.

`a_func_name`**:** (Optional) A name to add to the id of a_db.

Returns:

a_db: The DB with one row of max values, with selected dimension replaced by the output of the given function. n: (Optional) Numbers of used values in each call of a_func.

Example:

```
a_db = tests_3D_db(rand(5, 5, 5));
» b_db = processDimNonNaNInf(a_db, 1, 'mean')
will find the mean of rows in each page of the random 3D matrix.
» b_db = processDimNonNaNInf(a_db, 1, @(x,y)(max(x, [], y)), 'max')
more complex function form with 'max'.
```

See also: `max` (p. **??**), `mean` (p. **??**), `feval` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2008/05/27

A.47.38 Method `tests_db/factoran`

Summary: Generates a database of factor loadings obtained from the factor analysis of db with factoran. Each row corresponds to a rotated factor and columns represent observed variables.

Usage:

```
a_factors_db = factoran(db, num_factors, props)
```

Description: Uses the promax method to rotate common factors.

Parameters:

`db`**:** A tests_db object.

`num_factors`**:** Number of common factors to look for.

`props`**:** A structure with any optional properties.

Returns:

a_factors_db: A corrcoefs_db of the coefficients and page indices.

See also: `tests_db` (p. 227), `corrcoefs_db` (p. 82)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/08

A.47.39 Method `tests_db/plotrows`

Summary: Creates a plot_stack describing the db rows.

Usage:

```
a_plot = plotrows(a_tests_db, axis_limits, orientation, props)
```

Parameters:

`a_tests_db`**:** A tests_db object.

`axis_limits`**:** If given, all plots contained will have these axis limits.

`orientation`**:** Stack orientation 'x' for horizontal, 'y' for vertical, etc.

`title_str`**:** Optional title string.

`props`**:** A structure with any optional properties passed to plot_stack.

Returns:

a_plot: A plot_stack object that can be plotted.

See also: `plot_abstract` (p. 157), `plotFigure` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/09

A.47.40 Method `tests_db/mean`

Summary: Returns the mean of the data matrix of a_db. Ignores NaN values.

Usage:

```
[a_db, n] = mean(a_db, dim)
```

Description: Does a recursive operation over dimensions in order to remove NaN values. This takes more time than a straightforward mean operation.

Parameters:

`a_db`**:** A tests_db object.

`dim`**:** Work down dimension.

Returns:

a_db: The DB with one row of mean values. n: (Optional) Numbers of non-NaN rows included in calculating each column.

See also: `mean` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/06

A.47.41 Method `tests_db/rankMatching`

Summary: Create a ranking db of row distances of db to given criterion db.

Usage:

```
a_ranked_db = rankMatching(db, crit_db, props)
```

Description: crit_db can be created with the matchingRow method. TestWeights modify the importance of each measure.

Parameters:

`db`**:** A tests_db to rank.

`crit_db`**:** A tests_db object holding the match criterion tests and stds.

`props`**:** A structure with any optional properties.

`limitSTD`**:** limit any measure to this many STDs max.

`tolerateNaNs`**:** If 0, rows with any NaN values are skipped
, if 1, NaN values are given a fixed 3xSTD penalty (default=1).

`testWeights`**:** Structure array associating tests and multiplicative weights.

`restoreWeights`**:** Reverse the testWeights application after calculating distances.

`topRows`**:** If given, only return this many of the top rows.

`useMahal`**:** Use the Mahalonobis distance from the covariance matrix in crit_db.

Returns:

a_ranked_db: A ranked_db object.

See also: `matchingRow` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/12/08

A.47.42 Method `tests_db/onlyRowsTests`

Summary: Returns a tests_db that only contains the desired tests and rows (and pages).

Usage:

```
obj = onlyRowsTests(obj, rows, tests, pages)
```

Description: Selects the given dimensions and returns in a new tests_db object.

Parameters:

`obj`**:** A tests_db object.

`rows, tests`**:** A logical or index vector of rows, or cell array of names of rows. If ':', all rows. For names, regular expressions are supported if quoted with slashes (e.g., '/a.*/'). See tests2idx.

`pages`**:** (Optional) A logical or index vector of pages. ':' for all pages.

Returns:

obj: The new tests_db object.

See also: `subsref` (p. **??**), `tests_db` (p. 227), `tests2idx` (p. **??**), `regexp` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/17

A.47.43 Method `tests_db/times`

Usage:

```
a_db = mtimes(left_obj, right_obj)
```

Parameters:

`left_obj, right_obj`**:** Operands of the multiplication. One or more must be of type tests_db.

Returns:

a_db: The resulting tests_db.

See also: `tests_db/times` (p. 250), `mtimes` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/05/24

A.47.44 Method `tests_db/renameColumns`

Summary: Rename an existing column or columns.

Usage:

```
a_db = renameColumns(a_db, test_names, new_names)
```

Description: This is a cheap operation than modifies meta-data kept in object.

Parameters:

`a_db`: A tests_db object.

`test_names`: A cell array of existing test names.

`new_names`: New names to replace existing ones.

Returns:

a_db: The tests_db object that includes the new columns.

Example:

```
» new_db = renameColumns(a_db, 'PulseIni100msSpikeRateISI_D40pA', 'Firing_rate');
» new_db = renameColumns(a_db, 'a', 'b', 'c', 'd');
```

See also: `allocateRows` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/09/30

A.47.45 Method `tests_db/statsAll`

Summary: Makes a stats_db with rows of mean, STD, SE, and CV of the tests' distributions in db.

Usage:

```
a_stats_db = statsAll(db, tests, props)
```

Parameters:

`db`: A tests_db object.

`tests`: A selection of tests (see onlyRowsTests).

`props`: A structure with any optional properties for stats_db.

Returns:

a_stats_db: A stats_db object.

See also: `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/08/24

A.47.46 Method `tests_db/getColNames`

Summary: Gets column names.

Usage:

```
col_names = getColNames(db, tests)
```

Description: Performs a light operation without touching the data matrix.

Parameters:

`db`: A tests_db object.

`tests`: Columns for which to get names (Optional, default = ':')

Returns:

col_names: A cell array of strings.

See also: `getColNames` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/05/24

A.47.47 Method `tests_db/addLastRow`

Summary: Inserts a row of observations at the end of tests_db.

Usage:

```
index = addLastRow(obj, row)
```

Description: Adds a new set of observations to the database and returns its row index. This operation is expensive because the whole database matrix needs to be duplicated and resized in order to add a single new row. The method of allocating a matrix, filling it up, and then providing it to the tests_db constructor is the preferred method of creating tests_db objects.

Parameters:

`obj`: A tests_db object.

`row`: A row vector that contains values for each DB column.

Returns:

obj: The tests_db object that includes the new row.

See also: `allocateRows` (p. **??**), `addRow` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/08

A.47.48 Method `tests_db/isinf`

Summary: Returns logical row indices of Inf-valued columns.

Usage:

```
rows = isinf(db, col)
```

Parameters:

`db`: A tests_db object.

`col`: Column to check (Optional, default = 1)

Returns:

rows: A logical column vector of rows.

See also: `isinf` (p. ??), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/08/16

A.47.49 Method `tests_db/subsref`

Summary: Defines indexing for tests_db objects for () and . operations.

Usage:

```
obj = obj(rows, tests) obj = obj.attribute
```

Description: Returns attributes or selects the given test columns and rows and returns in a new tests_db object.

Parameters:

`obj`: A tests_db object.

`rows`: A logical or index vector of rows. If ':', all rows.

`tests`: Cell array of test names or column indices. If ':', all tests.

`attribute`: A tests_db class attribute.

Returns:

obj: The new tests_db object.

See also: `subsref` (p. ??), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/17

A.47.50 Method tests_db/eq

Summary: Equality (==) operator. Returns logical indices of db rows that match with given row.

Usage:

```
rows = eq(db, row)
```

Parameters:

db: A tests_db object.

row: Row array to be compared with db rows.

Returns:

rows: A logical or index vector to be used in indexing db objects.

See also: eq (p. **??**), tests_db (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/17

A.47.51 Method tests_db/plotBox

Summary: Creates a boxplot from each column in tests_db in separate axes.

Usage:

```
a_plot = plotBox(a_tests_db, title_str, props)
```

Parameters:

a_tests_db: A tests_db object.

title_str: Optional title.

props: A structure with any optional properties.

- putLabels: Put special column name labels.
- notch: If 1, put notches on boxplots (default=1).
- whis: Whisker size passed to boxplotp (default=1.5);

Returns:

a_plot: A plot_abstract object that can be plotted.

See also: plot_abstract (p. 157), plotFigure (p. **??**), boxplotp (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2008/01/16

A.47.52 Method tests_db/uminus

Summary: Unary minus or negation.

Usage:

```
a_db = uminus(left_obj)
```

Parameters:

`left_obj`: A tests_db object.

Returns:

a_db: The resulting tests_db.

See also: uminus (p. **??**), uop (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2008/01/16

A.47.53 Method tests_db/subsasgn

Summary: Defines generic index-based assignment for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/02/06

A.47.54 Method tests_db/dbsize

Summary: Returns the size of the data matrix of db.

Usage:

```
s = dbsize(db)
```

Parameters:

`db`: A tests_db object.

Returns:

s: The size values.

See also: size (p. **??**), tests_db (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/06

A.47.55 Method `tests_db/sqrt`

Summary: Takes the square root of a_db.

Usage:

```
a_db = sqrt(a_db)
```

Description: Overloaded sqrt function.

Parameters:

`a_db`: A tests_db.

Returns:

a_db: The resulting tests_db.

See also: `sqrt` (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/12/13

A.47.56 Method `tests_db/plotTestsHistsMatrix`

Summary: Create a matrix plot of test histograms.

Usage:

```
a_pm = plotTestsHistsMatrix(a_db, title_str, props)
```

Description: Skips the 'ItemIndex' test.

Parameters:

`a_db`: A params_tests_db object.

`title_str`: (Optional) A string to be concatanated to the title.

`props`: A structure with any optional properties, passed to plot_abstract.

- `orient`: Orientation of the plot_stack. 'x', 'y', or 'matrix' (default).
- `histBins`: Number of histogram bins.
- `quiet`: Don't put the DB id on the title.
- `axisLimits`: Only x-ranges are used from this expression.

Returns:

a_pm: A plot_stack with the plots organized in matrix form

See also: `params_tests_profile` (p. 140), `plotVar` (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/17

A.47.57 Method `tests_db/anyRows`

Summary: Returns db rows matching any of the given rows.

Usage:

```
idx = anyRows(db, rows)
```

Description: The db rows are compared to each row and row indices succeeding any of these comparisons are returned.

Parameters:

`db`: A tests_db object.

`rows`: Row array, matrix or database to be compared with db rows.

Returns:

idx: A logical column vector of matching db row indices. rows_idx: Indices of rows entries corresponding to each db row. Non-matching entries were left as NaN.

Example:

```
» db(anyRows(db(:, 'trial') == [12; 46; 37]), :)
returns a db with rows having trial equal to any of the given values.
```

See also: `compareRows` (p. **??**), `eq` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/17

A.47.58 Method `tests_db/gt`

Summary: Greater than (>) operator. Returns logical indices of db rows that are greater than given row.

Usage:

```
rows = gt(db, row)
```

Parameters:

`db`: A tests_db object.

`row`: Row array to be compared with db rows.

Returns:

rows: A logical or index vector to be used in indexing db objects.

See also: `gt` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/17

A.47.59 Method `tests_db/hist3`

Summary: Returns a 2D histogram of chosen two database columns.

Usage:

```
a_histogram_db = hist3(db, col1, col2, num_bins, props)
```

Description: Generates a histogram_db object with rows corresponding to histogram entries. If an array of DBs is given, finds and uses common histogram bin centers.

Parameters:

`db`**:** A tests_db object.

`col1, col2`**:** Columns to find the histogram.

`num_bins`**:** Two numbers for histogram bins (Optional, default=100), or Two-element cell of histogram bin centers vectors.

`props`**:** A structure with any optional properties.

`normalized`**:** If 1, normalize histogram counts.

Returns:

a_histogram_db: A histogram_db object containing the 2D histogram.

Example:

```
» a_hist_db = hist3(my_db, 'spike_width', 'spike_rate');
» plot(a_hist_db);
```

See also: `histogram_db` (p. 96), `tests_db` (p. 227), `hist3` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2008/04/02

A.47.60 Method `tests_db/kmeansCluster`

Summary: Generates a database of cluster centers obtained from a k-means cluster analysis with the command kmeans.

Usage:

```
a_cluster_db = kmeansCluster(db, num_clusters, props)
```

Parameters:

`db`**:** A tests_db object.

`num_clusters`**:** Number of clusters to form.

`props`**:** A structure with any optional properties.

`DistanceMeasure`**:** Choose one appropriate for kmeans.

Returns:

a_cluster_db: A tests_db where each row is a cluster center. a_hist_db: histogram_db showing cluster membership from original db. idx: Cluster indices of each row or original db. sum_distances: Quality of clustering indicated by total distance from centroid for each cluster.

See also: `tests_db` (p. 227), `histogram_db` (p. 96)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/04/06

A.47.61 Method `tests_db/compareRows`

Summary: Returns differing rows of db and the given row(s).

Usage:

```
[idx, compared] = compareRows(db, rows)
```

Description: It can compare all db rows to corresponding row entries or to a single row. For the case with only one entry, returns all db rows that do not match the given row in idx, and the result of the differences in compared. For the case of multiple rows, rows must have exactly same number of rows with db. In both cases, idx must be negated to test for equality.

Parameters:

`db`: A tests_db object.

`rows`: Row array, matrix or database to be compared with db rows.

Returns:

idx: A inverted logical column vector of comparison results. (false if db == rows, true otherwise) compared: A column vector of differences of each DB row to the given row (i.e., compared = db - rows).

Example:

```
» db(db(:, 'trial') > [12]), :)
calls gt which calls compareRows to check for equality.  Returns a db
only containing rows with trial numbers greater than 12.
```

See also: `eq` (p. **??**), `anyRows` (p. **??**), `ge` (p. **??**), `gt` (p. **??**), `le` (p. **??**), `lt` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/17

A.47.62 Method `tests_db/statsMeanStd`

Summary: Generates a stats_db object with mean, STD, and number of observations of the tests' distributions.

Usage:

```
a_stats_db = statsMeanStd(db, tests, props)
```

Parameters:

`db`: A tests_db object.

`tests`: A selection of tests (see onlyRowsTests).

`props`: A structure with any optional properties for stats_db.

Returns:

a_stats_db: A stats_db object.

See also: `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/07

A.47.63 Method `tests_db/display`

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.47.64 Method `tests_db/groupBy`

Summary: Groups same values of column(s) into separate pages of a 3D db.

Usage:

```
a_tests_3D_db = groupBy(db, cols)
```

Description: Functionality similar to SQL's GROUP BY keyword. This function uses invarValues.

Parameters:

`db`: A tests_db object.

`cols`: Columns whose same values will be in one page (see tests2cols for column representation).

Returns:

a_tests_3D_db: A tests_3D_db object of organized values.

Example:

```
» a_db = tests_db([ ...  ], 'par1', 'par2', 'measure1', 'measure2')
» a_3d_db = groupBy(a_db, 'par1')
» » joined_3d_db = joinRows(a_db, a_3d_db)
» displayRows(joined_3d_db(:, :, 1))
```

See also: `invarValues` (p. **??**), `tests_3D_db` (p. 220), `tests_3D_db/corrCoefs` (p. 225), `tests_3D_db/plotPair` (p. **??**), `joinRows` (p. **??**), `tests_3D_db/swapRowsPages` (p. 227), `tests_3D_db/mergePages` (p. 222)

Author: Cengiz Gunay <cgunay@emory.edu>, 2008/05/27

A.47.65 Method `tests_db/statsBounds`

Summary: Generates a stats_db object with three rows corresponding to the mean, min, max and number of observations of the tests' distributions.

Usage:

```
a_stats_db = statsBounds(a_db, tests, props)
```

Description: A page is generated for each page of data in db.

Parameters:

`a_db`**:** A tests_db object.

`tests`**:** A selection of tests (see onlyRowsTests).

`props`**:** A structure with any optional properties for stats_db.

Returns:

a_stats_db: A stats_db object.

See also: `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/07

A.47.66 Method `tests_db/end`

Summary: Overloaded primitive matlab function, returns maximal dimension size.

Usage:

```
s = end(db, index, total)
```

Parameters:

`db`**:** A tests_db object.

Returns:

s: The size.

See also: `size` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/06

A.47.67 Method `tests_db/isnanrows`

Summary: Finds rows with any NaN values. Returns logical indices of db rows.

Usage:

```
rows = isnanrows(db)
```

Description: Some operations need that no NaN values exist in the matrix. This method can be used to find and then remove NaN-contaminated rows from DB. Note that sometimes no rows can be found, and some columns should be discarded before this operation.

Parameters:

`db`**:** A tests_db object.

Returns:

rows: A logical vector to be used in indexing db objects or passed through other logical operators.

See also: `isnan` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/08

A.47.68 Method `tests_db/le`

Summary: Less or equal (<=) operator. Returns logical indices of db rows that are less than or equal to given row.

Usage:

```
rows = le(db, row)
```

Parameters:

`db`**:** A tests_db object.

`row`**:** Row array to be compared with db rows.

Returns:

rows: A logical or index vector to be used in indexing db objects.

See also: `le` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/17

A.47.69 Method `tests_db/plotUniquesStatsStacked3D`

Summary: Stack of 2D image plots of a column mean at unique values of three other columns.

Usage:

```
a_stacked_plot = plotUniquesStatsStacked3D(a_db, unique_test1, unique_test2, unique_test3,
stat_test, title_str, props)
```

Parameters:

`a_db`: A tests_db.

`unique_test1, unique_test2`: Columns whose unique values make up the X

& Y of the 2D image plot.

`unique_test3`: Column whose unique values make up stacked dimension.

`stat_test`: Column for which statsMeanSTD will be calculated for each unique value.

`props`: A structure with any optional properties.

(rest passed to plotUniquesStats2D and plot_stack).

Returns:

a_stacked_plot: A plot_abstract object to be plotted.

See also: `tests_db` (p. 227), `sortedUniqueValues` (p. **??**), `statsMeanStd` (p. **??**), `plot_abstract` (p. 157), `plotImage` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2008/04/15

A.47.70 Method `tests_db/plotScatter3D`

Summary: Create a 3D scatter plot of the given three tests.

Usage:

```
a_p = plotScatter3D(a_db, test1, test2, test3, title_str, short_title, props)
```

Parameters:

`a_db`: A params_tests_db object.

`test1, test2, test3`: X, Y, & Z variables.

`title_str`: (Optional) A string to be concatanated to the title.

`short_title`: (Optional) Few words that may appear in legends of multiplot.

`props`: A structure with any optional properties.

`LineStyle`: Plot line style to use. (default: 'x')

`Regress`: Calculate and plot a linear regression.

`quiet`: If 1, don't include database name on title.

Returns:

a_p: A plot_abstract.

See also:

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/11/30

A.47.71 Method `tests_db/rop`

Summary: Prepares aligned columns in two DBs or one DB and a scalar for an array arithmetic operation.

Usage:

```
a_db = rop(left_obj, right_obj, op_func, op_id)
```

Description: If DBs have mismatching columns only the common columns will be kept. In any case, the resulting DB columns will be sorted in the order of the left-hand-side DB. Array addition (plus), subtraction (minus), multiplication (mtimes) and division (rdivide) use this function to align columns.

Parameters:

`left_obj, right_obj`: Operands of the operation. One must be of type tests_db and the other can be a scalar or tests_db.

`op_func`: Operation function (e.g., @plus).

`op_id`: A string to represent the operation that will show up in the returned id.

Returns:

a_db: The resulting tests_db.

See also: `tests_db/plus` (p. 269), `tests_db/minus` (p. 241), `tests_db/mtimes` (p. 234), `tests_db/rdivide` (p. 235)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/12/13

A.47.72 Method tests_db/vertcat

Summary: Vertical concatanation [db;with_db;...] operator.

Usage:

```
a_db = vertcat(db, with_db, ...)
```

Description: Concatanates rows of with_db to rows of db. Overrides the built-in vertcat function that is called when [db;with_db] is executed. If the first argument is a array of DBs, then this functionality is not needed; built-in vertcat is called.

Parameters:

`db`: A tests_db object.

`with_db`: A tests_db object whose rows are concatanated to db.

Returns:

a_db: A tests_db that contains rows of db and with_db.

See also: `vertcat` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/01/25

A.47.73 Method tests_db/lt

Summary: Less than (<) operator. Returns logical indices of db rows that are less than given row.

Usage:

```
rows = lt(db, row)
```

Parameters:

`db`: A tests_db object.

`row`: Row array to be compared with db rows.

Returns:

rows: A logical or index vector to be used in indexing db objects.

See also: `lt` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/17

A.47.74 Method tests_db/unique

Summary: Returns DB with unique rows.

Usage:

```
a_db = unique(a_db)
```

Parameters:

a_db: tests_db from which to find uniques.

Returns:

a_db: The resulting tests_db.

See also: unique (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/11/19

A.47.75 Method tests_db/addRow

Summary: Inserts a row of observations to tests_db at the given row index.

Usage:

```
index = addRow(obj, row, index)
```

Description: Adds a new set of observations to the database and returns the new DB. This operation is expensive in the sense that the whole database matrix needs to be copied to be passed to this function just to add a single new row. The method of allocating a matrix, filling it up, and then providing it to the tests_db constructor is the preferred method of creating tests_db objects.

Parameters:

obj: A tests_db object.

row: A row vector that contains values for each DB column.

index: The row index.

Returns:

obj: The tests_db object that includes the new row.

See also: addLastRow (p. ??), allocateRows (p. ??), tests_db (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/08

A.47.76 Method `tests_db/plotYTests`

Summary: Create a plot given database measures against given X-axis values, for each row.

Usage:

```
a_p = plotYTests(a_db, x_vals, tests, axis_labels, title_str, short_title, command, props)
```

Parameters:

`a_db`: A params_tests_db object.

`x_vals`: A vector of X-axis values.

`tests`: A vector or cell array of columns to correspond to each value from x_vals.

`title_str`: (Optional) A string to be concatanated to the title.

`short_title`: (Optional) Few words that may appear in legends of multiplot.

`command`: (Optional) Command to do the plotting with (default: 'plot')

`props`: A structure with any optional properties.

`LineStyle`: Plot line style to use. (default: 'd-')
`ShowErrorbars`: If 1, errorbars are added to each point.
`StatsDB`: If given, use this stats_db for the errorbar (default=statsMeanStd(a_db)).
`quiet`: If 1, don't include database name on title.

Returns:

a_p: A plot_abstract.

Example:

```
» a_p = plotYTests(a_db_row, [0 40 100 200], ...
'IniSpontSpikeRateISI_0pA', 'PulseIni100msSpikeRateISI_D40pA', ...
'PulseIni100msSpikeRateISI_D100pA', 'PulseIni100msSpikeRateISI_D200pA', ...
'current pulse [pA]', 'firing rate [Hz]', ', f-I curves', 'neuron 1');
» plotFigure(a_p);
```

See also: `plotFigure` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/01/23

A.47.77 Method `tests_db/plotUniquesStatsBars`

Summary: Creates a mean-STD bar plot of a column for unique values of another column.

Usage:

```
a_bar_plot = plotUniquesStatsBars(a_db, unique_test, stat_test, title_str, props)
```

Parameters:

`a_db`**:** A tests_db.

`unique_test`**:** Column for which to generate bars for each of its unique values.

`stat_test`**:** Column for which statsMeanSTD will be calculated for each bar.

`props`**:** A structure with any optional properties.

`popMean`**:** If specified, plot a dotted line specifying the population mean. If NaN, calculated from a_db.

`yLims`**:** two-element vector for specifying y axis limits showing interesting part of the bar plot.

`uniqueVals`**:** Use these unique values for unique_test. (rest passed to plot_bars [and plot_superpose if popMean]).

Returns:

a_bar_plot: A plot_abstract object to be plotted.

Example:

```
» plotFigure(plotUniquesStatsBars(triplet_param_success_db, 'F_tau_m', ...
'successDefault', 'across triplets', ...
struct('fixedSize', [4 2], 'yLims', [.7 .9], ...
'popMean', 0.82, 'quiet', 1)))
```

See also: `tests_db` (p. 227), `sortedUniqueValues` (p. **??**), `statsMeanStd` (p. **??**), `plot_bars` (p. 163)

Author: Cengiz Gunay <cgunay@emory.edu>, 2008/04/14

A.47.78 Method `tests_db/plotScatter`

Summary: Create a scatter plot of the given two tests.

Usage:

```
a_p = plotScatter(a_db, test1, test2, title_str, short_title, props)
```

Parameters:

`a_db`**:** A params_tests_db object.

`test1, test2`**:** X & Y variables.

`title_str`**:** (Optional) A string to be concatanated to the title.

`short_title`**:** (Optional) Few words that may appear in legends of multiplot.

`props`**:** A structure with any optional properties.

`LineStyle`**:** Plot line style to use. (default: 'x')

`Regress`**:** Calculate and plot a linear regression.

`quiet`: If 1, don't include database name on title.

Returns:

a_p: A plot_abstract.

See also:

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/09/29

A.47.79 Method `tests_db/assignRowsTests`

Summary: Assign the values to the tests and rows (and pages) of the tests_db.

Usage:

```
obj = assignRowsTests(obj, val, rows, tests, pages)
```

Description: Selects the given dimensions and returns in a new tests_db object.

Parameters:

`obj`: A tests_db object.

`val`: DB object or data matrix to be assigned to the addressed indices.

`rows`: A logical or index vector of rows. If ':', all rows.

`tests`: Cell array of test names or column indices. If ':', all tests.

`pages`: (Optional) A logical or index vector of pages. ':' for all pages.

Returns:

obj: The new tests_db object.

See also: `subsref` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/02/08

A.47.80 Method `tests_db/plus`

Summary: Adds a DB to another or to a scalar.

Usage:

```
a_db = plus(left_obj, right_obj)
```

Description: If DBs have mismatching columns only the common columns will be kept. In any case, the resulting DB columns will be sorted in the order of the left-hand-side DB.

Parameters:

`left_obj, right_obj`**:** Operands of the addition. One must be of type tests_db and the other can be a scalar or tests_db.

Returns:

a_db: The resulting tests_db.

See also: `plus` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/12/13

A.47.81 Method `tests_db/displayRows`

Summary: Displays rows of data with associated column labels.

Usage:

```
s = displayRows(db, rows, pages)
```

Parameters:

`db`**:** A tests_db object.

`rows`**:** Indices of rows in db.

`pages`**:** Pages of db.

Returns:

s: A cell array of trasposed database contents, prefixed with column names on each row. Meant to be displayed on the screen.

See also: `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/15

A.47.82 Method `tests_db/testsHists`

Summary: Calculates histograms for all tests.

Usage:

```
t_hists = testsHists(a_db, num_bins)
```

Parameters:

`a_db`**:** A tests_db object.

`num_bins`**:** Number of histogram bins (Optional, default=100), or vector of histogram bin centers.

Returns:

t_hists: An array of histograms for each test in a_db.

See also: `params_tests_profile` (p. 140)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/04/27

A.47.83 Method `tests_db/ge`

Summary: Greater or equal to (>=) operator. Returns logical indices of db rows that are greater than or equal to given row.

Usage:

```
rows = ge(db, row)
```

Parameters:

`db`**:** A tests_db object.

`row`**:** Row array to be compared with db rows.

Returns:

rows: A logical or index vector to be used in indexing db objects.

See also: `ge` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/17

A.47.84 Method `tests_db/swapRowsPages`

Summary: Swaps the row dimension with the page dimension of the tests_db.

Usage:

```
a_db = swapRowsPages(db)
```

Description: Watered-down version of the tests_3D_db/swapRowsPages function that does not touch row indices.

Parameters:

`db`**:** A tests_db object.

Returns:

a_db: A tests_db object.

See also: `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/04

A.47.85 Method `tests_db/plotUniquesStats2D`

Summary: 2D image plot of the change in column mean for unique values of two other columns.

Usage:

```
an_image_plot = plotUniquesStats2D(a_db, unique_test1, unique_test2, stat_test, title_str,
props)
```

Parameters:

- `a_db`**:** A tests_db.
- `unique_test1, unique_test2`**:** Columns whose unique values make up the X

  & Y of the 2D image plot.
- `stat_test`**:** Column for which statsMeanSTD will be calculated for each unique value.
- `props`**:** A structure with any optional properties.
  - `popMean`**:** If specified, plot a dotted line specifying the population mean. If NaN, calculate from given a_db.
  - `popDev`**:** Use this value +/- to choose colorbar extents (default=.3 or 2*STD if popMean=NaN).
  - `colorbar`**:** Show vertical colorbar axis (see plotImage).
  - `uniqueVals1,uniqueVals2`**:** Use these unique values for unique_test1,unique_test2.
  - `statsFunc`**:** tests_db/stats* method to use (default: statsMeanStd).
  - `statsRow`**:** The row to pick from statsFunc results (default: mean). (rest passed to plotImage and plot_abstract).

Returns:

an_image_plot: A plot_abstract object to be plotted.

Example:

```
» plotFigure(plotUniquesStats2D(triplet_param_success_db, ...
'F_tau_m', 'S_tau_m', 'successDefault', ...
'accross triplets',
struct('fixedSize', [4 3], 'popMean', NaN, ...
'colorbar', 1, 'quiet', 1, 'border', [0.03 0 0.03 0])))
```

See also: `tests_db` (p. 227), `sortedUniqueValues` (p. **??**), `statsMeanStd` (p. **??**), `plot_abstract` (p. 157), `plotImage` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2008/04/14

A.47.86 Method `tests_db/plot_bars`

Summary: Creates a bar graph comparing all DB rows in groups, with a separate axis for each column.

Usage:

```
a_plot = plot_bars(a_tests_db, title_str, props)
```

Parameters:

`a_tests_db`**:** A tests_db object.

`title_str`**:** (Optional) The plot title.

`props`**:** A structure with any optional properties.

Returns:

a_plot: A object of plot_bars or one of its subclasses.

See also: `plot_abstract` (p. 157), `plot_simple` (p. 169)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/03/13

A.47.87 Method `tests_db/diff`

Summary: Creates a tests_db by taking the derivative of all tests.

Usage:

```
a_db = diff(a_db, props)
```

Description: Applies the diff function to whole DB. The resulting DB will have one less row.

Parameters:

`a_db`**:** A tests_db object.

`props`**:** Optional properties.

Returns:

a_db: The resulting tests_db.

See also: `diff` (p. **??**), `tests_3D_db/getDiff2DDB` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/05/24

A.47.88 Method `tests_db/delColumns`

Summary: Deletes columns from tests_db.

Usage:

```
obj = delColumns(obj, tests)
```

Description: Deletes test columns from the database and returns the new DB. This operation is expensive in the sense that the whole database matrix needs to be copied just to delete a single column. The method of allocating a matrix, filling it up, and then providing it to the tests_db constructor is the preferred method of creating tests_db objects. This method may be used for measures obtained by operating on raw measures.

Parameters:

`obj`**:** A tests_db object.

`tests`**:** Numbers or names of tests (see tests2cols)

Returns:

obj: The tests_db object that is missing the columns.

See also: `allocateRows` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/10/06

A.47.89 Method `tests_db/meanDuplicateRows`

Summary: Row-reduces a db by finding sets of rows with same main_cols values, and replacing each set with a single row containing main_cols and the mean of rest_cols.

Usage:

```
a_tests_db = meanDuplicateRows(db, main_cols, rest_cols)
```

Description: The database is sorted for the values of the columns of interest (main_cols) and all rows with duplicate values of these columns are identified. The rest of the columns (rest_cols) are averaged and reduced to a single row, and attached to the nominal values of main_cols. Two additional parameter columns will be added to the database created. The NumDuplicates column is the the number of duplicates used in the mean operation, and RowIndex is the row number points to the first of a set of duplicate values.

Parameters:

`db`**:** A tests_db object.

`main_cols`**:** Vector of columns in which to find duplicates.

`rest_cols`: Vector of columns to be averaged for duplicate main_cols.

Returns:

a_tests_db: The db object of with the means on page 1 and standard deviations on page 2.

See also: `tests_db/mean` (p. 248), `tests_db/std` (p. 246), `sortedUniqueValues` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/30

A.47.90 Method `tests_db/statsMeanSE`

Summary: Generates a stats_db object with two rows corresponding to the mean and standard error (SE) of the tests' distributions.

Usage:

```
a_stats_db = statsMeanSE(db, tests, props)
```

Parameters:

`db`: A tests_db object.

`tests`: A selection of tests (see onlyRowsTests).

`props`: A structure with any optional properties for stats_db.

Returns:

a_stats_db: A stats_db object.

See also: `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/07

A.47.91 Method `tests_db/sum`

Summary: Creates a tests_db by summing all rows.

Usage:

```
a_db = sum(a_db, dim, props)
```

Description: Applies the sum function to whole DB. The resulting DB will have one row.

Parameters:

`a_db`: A tests_db object.

`props`: Optional properties.

Returns:

a_db: The resulting tests_db.

See also: sum (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/05/24

A.47.92 Method tests_db/plot_abstract

Summary: Default visualization for a database.

Usage:

```
a_pm = plot_abstract(a_db, title_str)
```

Description: Calls plotTestsHistsMatrix. Subclasses should override this method to provide their own visualization.

Parameters:

a_db: A params_tests_db object.

title_str: (Optional) A string to be concatanated to the title.

props: A structure with any optional properties.

Returns:

a_pm: A plot_stack with the plots organized in matrix form

Example:

```
» plot(my_db, ': first impression')
will call this function and send the generated plot to the plotFigure function.
```

See also: plot_abstract/plot_abstract (p. 157), plotTestsHistsMatrix (p. ??), plotFigure (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/08/17

A.47.93 Method tests_db/crossProd

Summary: Create a DB by taking the cross product of two database row sets.

Usage:

```
cross_db = crossProd(a_db, b_db)
```

Description: This is not a vector cross product operation. Each row of the two DBs are matched and added as a new row to a DB. The end is a DB with all combinations of rows from both DBs. The final DB contains columns of both DBs.

Parameters:

`a_db, b_db`: A tests_db object.

Returns:

cross_db: The tests_db object with all combinations of rows.

See also: `allocateRows` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/10/11

A.47.94 Method `tests_db/tests2cols`

Summary: Find column numbers from a test names/numbers specification.

Usage:

```
cols = tests2cols(db, tests)
```

Description: Uses tests2idx.

Parameters:

`db`: A tests_db object.

`tests`: Either a single or array of column numbers, or a single test name or a cell array of test names. If ':', all tests. For name strings, regular expressions are supported if quoted with slashes (e.g., '/a.*/'). See tests2idx for more.

Returns:

cols: Array of column indices.

See also: `tests_db` (p. 227), `tests2idx` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/07

A.47.95 Method `tests_db/rows2Struct`

Summary: Convert given rows of database to a structure array.

Usage:

```
s = rows2Struct(db, rows, pages)
```

Parameters:

`db`: A tests_db object.

`rows`: Indices of rows in db.

`pages`**:** Pages of db.

Returns:

s: A structure of column name and value pairs.

See also: `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/08/17

A.47.96 Method `tests_db/max`

Summary: Returns the max of the data matrix of a_db. Ignores NaN and Inf values.

Usage:

```
[a_db, n] = max(a_db, dim)
```

Description: Does a recursive operation over dimensions in order to remove NaN and Inf values. This takes more time than a straightforward max operation.

Parameters:

`a_db`**:** A tests_db object.

`dim`**:** Work down dimension.

Returns:

a_db: The DB with one row of max values. n: (Optional) Numbers of non-NaN rows included in calculating each column.

See also: `max` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/06

A.48 Class `trace`

A.48.1 Constructor `trace/trace`

Summary: A generic trace from a cell. It can be voltage, current, etc.

Usage:

```
obj = trace(data_src, dt, dy, id, props)
```

Description: Traces for specific experimental or simulation protocols can extend this class for adding new parameters. This object is designed to recognize most data file formats. See the data_src parameter below.

Parameters:

`data_src`: Trace data as a column vector OR name of a data file generated by either

Genesis (.bin, .gbin, .genflac), PCDX (.all), or Matlab (.mat).

`dt`: Time resolution [s]

`dy`: y-axis resolution [ISI (V, A, etc.)]

`id`: Identification string

`props`: A structure with any optional properties.

- `scale_y`: Y-axis scale to be applied to loaded data.
- `offset_y`: Y-axis offset to be added to loaded and scaled data.
- `trace_time_start`: Samples in the beginning to discard [dt]
- `baseline`: Resting potential.
- `channel`: Channel to read from file Genesis, PCDX, or Neuron file.
- `file_type`: Specify file type instead of guessing from extension: 'genesis': Raw binary files created with Genesis disk_out method. 'genesis_flac': Compressed Genesis binary files. 'neuron': Binary files created with Neuron's Vector.vwrite method. 'pcdx': .ALL data acquisition files from PCDX program. 'matlab': Matlab .MAT binary files with matrix data.
- `traces`: Traces to read from PCDX file.
- `spike_finder`: Method of finding spikes (1 for findFilteredSpikes, 2 for findspikes).
- `threshold`: Spike threshold used in spike finders.
- `init_Vm_method`: Method of finding spike thresholds during spike shape calculation (see spike_shape/spike_shape).
- `init_threshold`: Spike initiation threshold (deriv or accel). (see above methods and implementation in calcInitVm)
- `init_lo_thr, init_hi_thr`: Low and high thresholds for slope.

`custom_filter`: Recommended if sampling rate differs appreciably from 10 kHz.

If custom_filter == 1, a filter with custom lowpass and highpass cutoffs can be specified. This allows for fast and accurate spike discrimination. The filter type used is a 2-pole butterworth, different than the default high-order Cheby2. Creates new prop called 'butterWorth' to hold the filter.

`lowPassFreq`: if set, it sets a new low pass cutoff. default is 3000Hz

`highPassFreq`: if set it sets a new high pass cutoff. default is 50 Hz

`quiet`: If 1, reduces the amount of textual description in plots, etc.

Returns a structure object with the following fields:

data: The trace column matrix. dt, dy, id, props (see above)

See also: `spikes` (p. 202), `spike_shape` (p. 189), `cip_trace` (p. 56), `period` (p. 140)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/07/30

A.48.2 Method trace/plotData

Summary: Plots a trace.

Usage:

```
a_plot = plotData(t, title_str, props)
```

Description: If t is a vector of traces, returns a vector of plot objects.

Parameters:

t: A trace object.

title_str: (Optional) String to append to plot title.

props: A structure with any optional properties.

timeScale: 's' for seconds, or 'ms' for milliseconds.

quiet: If 1, only display given title_str.

(rest passed to plot_abstract.)

Returns:

a_plot: A plot_abstract object that can be visualized.

See also: trace (p. 278), trace/plot (p. 280), plot_abstract (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/17

A.48.3 Method trace/get

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.48.4 Method trace/plot

Summary: Plots a trace.

Usage:

```
h = plot(t)
```

Parameters:

t: A trace object.

title_str: (Optional) String to append to plot title.

props: A structure with any optional properties, passed to plot_abstract.

Returns:

h: Handle to figure object.

See also: trace (p. 278), plot_abstract (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.48.5 Method `trace/setProp`

Summary: Generic method for setting optional object properties.

Usage:

```
obj = setProp(obj, prop1, val1, prop2, val2, ...)
```

Description: Modifies or adds property values. As many property name-value pairs can be specified.

Parameters:

`obj`**:** Any object that has a props field.

`attr`**:** Property name

`val`**:** Property value.

Returns:

obj: The new object with the updated properties.

See also:

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/22

A.48.6 Method `trace/calcMax`

Summary: Calculates the maximal value of the given period of the trace, t.

Usage:

```
[max_val, max_idx] = calcMax(t, period)
```

Parameters:

`t`**:** A trace object.

`period`**:** A period object (optional).

Returns:

max_val: The max value. max_idx: Its index in the trace.

See also: period (p. 140), trace (p. 278)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/07/30

A.48.7 Method trace/getResults

Summary: Runs all tests defined by this class and return them in a structure.

Usage:

```
results = getResults(t)
```

Parameters:

t: A trace object.

Returns:

results: A structure associating test names to values in ms and mV (or mA).

See also: spike_shape (p. 189)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/13

A.48.8 Method trace/calcAvg

Summary: Calculates the average value of the given period of the trace, t.

Usage:

```
avg_val = calcAvg(t, period)
```

Parameters:

t: A trace object.

period: A period object (optional).

See also: period (p. 140), trace (p. 278)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/07/30

A.48.9 Method trace/analyzeSpikesInPeriod

Summary: Returns results and a db of spikes by collecting test results of a cip_trace, analyzing each individual spike.

Usage:

```
[results period_spikes a_spikes_db spikes_stats_db spikes_hists_dbs] =
analyzeSpikesInPeriod(a_cip_trace, a_spikes, period, prefix_str)
```

Parameters:

a_cip_trace: A cip_trace object.

a_spikes: A spikes object from the a_cip_trace object.

`period`: A period of object of a_cip_trace object of interest.

`prefix_str`: Prefix string to be added to spike shape results.

Returns:

results: Results structure names prefixed with prefix_str. period_spikes: Corrected spikes object for this period. a_spikes_db: A mini spikes database of results from each spike in period. spikes_stats_db: Statistics from the mini spikes database. spikes_hists_dbs: Cell array of histograms from the mini spikes database.

See also: `cip_trace` (p. 56), `spikes` (p. 202), `period` (p. 140), `spike_shape` (p. 189), `getProfileAllSpikes` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/05/04

A.48.10 Method `trace/set`

Summary: Generic method for setting object attributes.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/08

A.48.11 Method `trace/periodWhole`

Summary: Returns the boundaries of the whole period of trace, t.

Usage:

```
whole_period = periodWhole(t)
```

Parameters:

`t`: A trace object.

See also: `period` (p. 140), `trace` (p. 278)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/07/30

A.48.12 Method `trace/subsref`

Summary: Defines generic indexing for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.48.13 Method `trace/calcMin`

Summary: Calculates the minimal value of the given period of the trace, t.

Usage:

```
[min_val, min_idx] = calcMin(t, a_period)
```

Parameters:

`t`**:** A trace object.

`a_period`**:** A period object (optional).

Returns:

min_val: The min value. min_idx: Its index in the trace.

See also: `period` (p. 140), `trace` (p. 278)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/07/30

A.48.14 Method `trace/runAvg`

Summary: Returns a trace which is the running average of this trace.

Usage:

```
avg_t = runAvg(t)
```

Parameters:

`t`**:** A trace object.

Returns:

avg_t: A trace object that contains the running average of this trace.

See also: `trace` (p. 278)

Author: Cengiz Gunay <cgunay@emory.edu>, 2008/05/14

A.48.15 Method `trace/display`

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.48.16 Method trace/getDy

Summary: Returns dy.

Usage:

```
dy = getDy(t)
```

Parameters:

t: A trace object.

Returns:

dy: The dy value.

See also: trace (p. 278)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/31

A.48.17 Method trace/findFilteredSpikes

Summary: Runs a frequency filter over the data and then finds all peaks using findspikes.

Usage:

```
[times, peaks, n] = findFilteredSpikes(t, a_period, plotit, minamp)
```

Description: Runs a 50-300 Hz band-pass filter over the data and then calls findspikes. The filter both removes low-frequency offset changes, such as cip period effects, and high-frequency noise that is detected as local peaks by findspikes. The spikes found are post-processed to make sure the rise and fall times are consistent. Note: The filter employed only works with data sampled at 10kHz.

Parameters:

t: Trace object

a_period: Period of interest.

plotit: Plots the spikes found if 1.

minamp (optional): minimum amplitude above baseline that must be reached. –> adjust as necessary to discriminate spikes from EPSPs.

Returns:

times: The times of spikes [dt].

peaks: The peaks corresponding to the times of spikes.

n: The number of spikes.

See also: findspikes (p. **??**), period (p. 140)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/03/08

A.48.18 Method trace/withinPeriod

Summary: Returns a trace object valid only within the given period.

Usage:

```
obj = withinPeriod(t, a_period, props)
```

Parameters:

t: A trace object.

a_period: The desired period

props: A structure with any optional properties.

useAvailable: If 1, don't stop if period not available, use maximum available.

Returns:

obj: A trace object

See also: trace (p. 278), period (p. 140)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/25

A.48.19 Method trace/spikes

Summary: Convert trace to spikes object for spike timing calculations.

Usage:

```
obj = spikes(trace, a_period, plotit, minamp)
```

Description: Creates a spikes object.

Parameters:

trace: A trace object.

a_period: A period object denoting the part of trace of interest
(optional, if empty vector, taken as wholePeriod).

plotit: If non-zero, a plot is generated for showing spikes found
(optional).

minamp: minimum amplitude that must be reached if using findFilteredSpikes.
–> adjust as needed to discriminate spikes from EPSPs. (optional)

See also: spikes (p. 202)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/07/30

A.48.20 Method trace/getSpike

Summary: Convert a spike in the trace to a spike_shape object.

Usage:

```
obj = getSpike(trace, spikes, spike_num, props)
```

Description: Creates a spike_shape object from desired spike. It is more efficient if you already have the spikes object.

Parameters:

`trace`**:** A trace object.

`spikes`**:** (Optional) A spikes object obtained from trace, calculated automatically if given as [].

`spike_num`**:** The index of spike to extract.

`props`**:** A structure with any optional properties.

`spike_id`**:** A prefix string added to the spike_shape object's id.

Example:

```
This will create an annotated plot of the third spike in my_trace:
» plotFigure(plotResults(getSpike(my_trace, [], 3)))
```

See also: `spike_shape` (p. 189)

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/04/19

A.48.21 Method trace/plot_abstract

Summary: Plots a trace by calling plotData.

Usage:

```
a_plot = plot_abstract(t, title_str, props)
```

Description: If t is a vector of traces, returns a vector of plot objects.

Parameters:

`t`**:** A trace object.

`title_str`**:** (Optional) String to append to plot title.

`props`**:** A structure with any optional properties.

`timeScale`**:** 's' for seconds, or 'ms' for milliseconds. (rest passed to plot_abstract.)

Returns:

a_plot: A plot_abstract object that can be visualized.

See also: trace (p. 278), trace/plot (p. 280), plot_abstract (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/17

A.48.22 Method trace/spike_shape

Summary: Convert averaged spikes in the trace to a spike_shape object.

Usage:

```
obj = spike_shape(trace, spikes, props)
```

Description: Creates a spike_shape object.

Parameters:

`trace`**:** A trace object.

`spikes`**:** A spikes object on trace.

See also: spike_shape (p. 189)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/04

A.49 Class trace_profile

A.49.1 Constructor trace_profile/trace_profile

Summary: Creates and collects test results of a trace.

Description: The first usage is fully customizable to be used from subclass constructors. The second usage generates the spikes and spike_shape objects, and collects some generic test results from them. This usage is only provided as an example and is not used practically.

Parameters:

`data_src`**:** The trace column OR the .MAT filename.

`dt`**:** Time resolution [s]

`dy`**:** y-axis resolution [ISI (V, A, etc.)]

`props`**:** See trace object.

Returns a structure object with the following fields:

trace, spikes, spike_shape, results, id, props.

See also: trace (p. 278), spikes (p. 202), spike_shape (p. 189)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/13

A.49.2 Method trace_profile/get

Summary: Defines generic attribute retrieval for objects.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/14

A.50 Utility functions

A.50.1 Function functions/TeXtable

Summary: Creates the LaTeX string for a floating table containing given contents.

Usage:

```
tex_string = TeXtable(contents, caption, props)
```

Parameters:

`contents`**:** Table contents in LaTeX.

`caption`**:** Table caption.

`props`**:** A structure with any optional properties.

- `rotate`**:** Degrees to rotate.
- `width`**:** Resize to this width.
- `height`**:** Resize to this height
- `center`**:** Align to center.
- `shortCaption`**:** Short version of caption to appear at list of tables.
- `floatType`**:** LaTeX float to use (default='table').
- `label`**:** Used for internal LaTeX references.

Returns:

tex_string: LaTeX string for table float.

See also:

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/12/13

A.50.2 Function functions/logLevels

Summary: Returns a logarithmic-scaled series between min_val and max_val with num_levels elements.

Usage:

```
levels = logLevels(min_val, max_val, num_levels)
```

Parameters:

min_val, max_val: The low and high boundaries for the output value.

num_levels: Number of elements to produce, including the boundaries.

Returns:

levels: A column vector of logarithmic series between min_val and max_val.

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/04/18

A.50.3 Function functions/parseGenesisFilename

Summary: Parses the GENESIS filename to get names and values of simulation parameters. Usage: names_vals = parseGenesisFilename(filename)

Description: Parameters: filename: GENESIS filename (no need to exist)

Returns:

names_vals: A two-column cell array with names and values.

See also: cip_profile (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/03/10

A.50.4 Function functions/gettracelist2

Summary: Gets a list of the form: '1 3 7-10' and returns a column vector with the traces numbers, and the number of traces.

Usage:

```
[traces, ntraces] = gettracelist2('list');
```

Description: Please note the space between single traces and the dash for ranges of traces.

Author: <adelgado@biology.emory.edu>

A.50.5 Function functions/loadtraces

Parameters:

file: PCDX file.

tracelist: A string of trace description, such as '1-10'.

channel: Channel to read from.

quiet: (Optional) If 1, produce on print outs.

Author: <adelgado@biology.emory.edu>

A.50.6 Function functions/properTeXLabel

Summary: Replaces characters in string or cell array of strings to make it valid in TeX documents.

Usage:

```
a_label = properTeXLabel( a_label )
```

Description: It will replace characters like space, '/', '.', etc.

Parameters:

a_label**:** A label string.

Returns:

a_label: The corrected proper a_label.

Example:

```
» a_label = properTeXLabel('this_day')
ans = 'this\_day'
```

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/01/17

A.50.7 Function functions/defaultValue

Summary: If variable unset (either nonexistent or empty), assign it a default value. Otherwise the variable remains unchanged.

Usage:

```
var = defaultValue(varname, a_defaultvalue)
```

Description: If the variable has already been defined, it keeps unchanged. If the variable doesn't exist or is an empty matrix, it will be assigned a default value to it.

Parameters:

varname**:** a string. the name of the variable.

a_defaultvalue**:** value for the variable.

Example:

```
SamplingRate = defaultValue('SamplingRate', 10);
```

Author: Li, Su

A.50.8 Function `functions/ns_CIPlist`

Author: Dawid Kurzyniec

A.50.9 Function `functions/diff3T`

Summary: Estimate of third derivative using Taylor expansion.

Usage:

```
deriv3 = diff3T(x, dy)
```

Description: d^3 x x(k-3) - 8 * x(k-2) + 13 * x(k-1) - 13 * x(k+1) + 8 * x(k+2) -
x(k+3) ——- = ———————————————————————— dy^3
8 * dy^3

Parameters:

`x`**:** A vector of x = f(y).

`dy`**:** The resolution of the discrete points in the vector.

Returns:

deriv3: Estimate of the derivative.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/15

A.50.10 Function `functions/findspikes_old`

Author: <adelgado@biology.emory.edu>, 2003-03-31

A.50.11 Function `functions/properTeXFilename`

Summary: Replaces characters in string to make it a valid filename for inclusion in TeX documents.

Usage:

```
filename = properTeXFilename( filename )
```

Description: It will replace characters like space, '/', '.', etc.

Parameters:

`filename`**:** An input filename string (without extension!).

Returns:

filename: The corrected proper filename.

Example:

```
» fname = properTeXFilename('hello world/1')
ans = 'hello_world+1'
```

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/12/20

A.50.12 Function `functions/mergeStructs`

Summary: Merges all the structures given as arguments and makes a single structure.

Usage:

```
results = mergeStructs( struct1 [, struct2, ...]  )
```

Description: The fields will in earlier arguments will have priority. So, while merging two structs, if there are duplicate fields, the fields in the first will be preserved.

Parameters:

struct(n): A structure.

Returns:

results: The merged structure.

Example:

```
mergeStructs( struct('hello', 1), struct('bye', 2) );
=> struct('hello', 1, 'bye', 2)
```

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/13

A.50.13 Function `functions/fillederrorbar`

Summary: Plots an errorbar with the middle points filled with the pen color.

Usage:

```
handles = fillederrorbar(...)
```

Parameters:

(see errorbar)

Returns:

handles: Handles to graphics objects.

See also:

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/13

A.50.14 Function `functions/struct2DB`

Summary: Converts a structure array to a tests_db object.

Usage:

```
a_tests_db = struct2DB(a_struct, props)
```

Description: Field names become column names in the DB.

Parameters:

`a_struct`**:** A structure to convert.

`props`**:** A structure with any optional properties, passed to tests_db.

Returns:

a_tests_db: A tests_db object.

See also:

Author: Cengiz Gunay <cgunay@emory.edu>, 2008/01/11

A.50.15 Function `functions/getfuzzyfield`

Author: Li, Su - 2007

A.50.16 Function `functions/maxima`

Summary: Find all local maxima.

Usage:

```
x_idx = maxima(x)
```

Description: Finds derivative sign-flipping points where the second derivative is less than zero.

Parameters:

`x`**:** A vector.

Returns:

x_idx: Indices of maxima.

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/04/18

A.50.17 Function functions/chanTables2DB

Summary: Creates a DB with channel tables exported from Genesis.

Usage:

```
a_chans_db = chanTables2DB(tables, id, props)
```

Parameters:

`tables`: Structures returned from the dump files generated by dump_chans.g.

`id`: String that identify the source of the tables structure.

`props`: A structure with any optional properties.
(rest passed to tests_db.)

Returns:

a_chans_db: A tests_db object containing channel tables.

See also: `trace` (p. 278), `trace/plot` (p. 280), `plot_abstract` (p. 157), `GP/common/dump_chans.g (Genesis)` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/03/07

A.50.18 Function functions/prefixStruct

Summary: Adds the given prefix to each of the field names in the structure.

Usage:

```
new_struct = prefixStruct(a_struct, prefix_str)
```

Parameters:

`a_struct`: A structure.

`prefix_str`: A string to be prefixed to each field name.

Returns:

new_struct: The new structure.

Example:

```
prefixStruct( struct('bye', 1), 'hello');
=> struct('hellobye', 1)
```

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/12/22

A.50.19 Function functions/array2str

Example:

```
string = array2str([1 2 3 4 6 8 10 12 13])
the output is '[1:4 6:2:12 13]'
```

A.50.20 Function functions/cell2TeX

Summary: Creates LaTeX string of a formatted table with the cell array's contents.

Usage:

```
tex_string = cell2TeX(a_cell, props)
```

Parameters:

- **a_cell:** A cell matrix to be tabularized.
- **props:** A structure with any optional properties.
  - **hasTitleRow:** The first row contains titles.
  - **hasTitleCol:** The first column contains titles.

Returns:

tex_string: LaTeX formatted table string.

See also:

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/12/09

A.50.21 Function functions/setAxisNonNaN

Summary: Returns the limits of the current axis replaced with the non-NaN elements of the given vector.

Usage:

```
new_axis = setAxisNonNaN(layout_axis)
```

Parameters:

- **layout_axis:** The axis position to layout this plot.

Returns:

new_axis: Modified axis.

Example:

```
» axis(setAxisNonNaN([0 100 NaN NaN]))
```

See also: plot_abstract (p. 157)

Author: Cengiz Gunay <cgunay@emory.edu>, 2007/10/29

A.50.22 Function `functions/diff2T_h4`

Summary: Estimate of second derivative using Taylor expansion (derived with same method as diffT).

Usage:

```
deriv2 = diff2T_h4(x, dy)
```

Description: d^2 x x(k-2) - x(k-1) - x(k+1) + x(k+2) ——- = ———————————— dy^2 6 * dy^2

Parameters:

`x`: A vector of x = f(y).

`dy`: The resolution of the discrete points in the vector.

Returns:

deriv2: Estimate of the derivative.

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/04/15

A.50.23 Function `functions/plotImage`

Summary: Function that plots a color matrix on current figure.

Usage:

```
h = plotImage(image_data, a_colormap, num_colors, props)
```

Parameters:

`image_data`: 2D matrix with image data.

`a_colormap`: Colormap vector, function name or handle to colormap (e.g., 'jet').

`num_colors`: Parameter to be passed to the a_colormap.

`props`: A structure with any optional properties.

- `colorbar`: If defined, show colorbar on plot.
- `colorbarProps`: Set colorbar axis properties.
- `colorbarLabel`: Set colorbar y-axis label.
- `truncateDecDigits`: Truncate labels to this many decimal digits.
- `maxValue`: Maximal value at num_colors to annotate the colorbar.
- `reverseYaxis`: If 1, display y-axis values in reverse (default=0).

Returns:

colors: RGB color matrix to be passed to colormap function.

See also: `colormap` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/06/05

A.50.24 Function `functions/diff2T`

Summary: Estimate of second derivative using Taylor expansion.

Usage:

```
deriv2 = diff2T(x, dy)
```

Description: d^2 x - x(k-2) + 16 * x(k-1) - 30 * x(k) + 16 * x(k+1) - x(k+2) ——- = ———————————————————————— dy^2 12 * dy^2

Parameters:

`x`: A vector of x = f(y).

`dy`: The resolution of the discrete points in the vector.

Returns:

deriv2: Estimate of the derivative.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/15

A.50.25 Function `functions/boxutilp`

Description: BOXUTILP(X) is a utility function for BOXPLOT, which calls BOXUTILP once for each column of its first argument. Use BOXPLOT rather than BOXUTILP.

A.50.26 Function `functions/ns_load_tracesets`

Summary: Return a set of physiol_cip_traceset objects loaded from a single NeuroSAGE HDF5 file.

Usage:

```
a_tss = ns_load_tracesets(data_src, props)
```

Description: This allows customized loading each NeuroSAGE file separately. Only loads traces that has the word 'cip' or 'spont' in the NeuroSAGE sequence name. Sample rate, channel gain and dy values are read from the acquisition data.

Parameters:

`data_src`: A pattern for one or more NeuroSAGE filename or structure output of ns_open_file.

`props`: A structure with any optional properties.

`VmChan`**:** (Optional) If a string, read voltage trace from channel having this string (e.g., 'Amp1 Vm'). If numeric, use as channel number. Added to the neuron_id to distinguish multiple neurons recorded in same file. If not specified, the first voltage channel is used.

`ImChan`**:** (Optional) Similar to VmChan for reading current trace. Does not affect neuron_id.

`VGain, IGain`**:** for HDF5 files, these two fields only works as a default value
when the gains are not specified in the file.

`IncludeSeq`**:** A string or cell array of strings specifying keywords in sequence name to look for.

`ExcludeSeq`**:** A string or cell array of strings specifying keywords in sequence name to avoid searching.

`addTreats`**:** Structure of default treatment names and their values for this traceset to keep consistent accross tracesets. Use only lowercase in treatment names.

`fixTreats`**:** Override wrong treatment information with these. Same format as addTreats.

`renameTreats`**:** Structure with from->to rename pairs.

`trials`**:** A vector of trials to load from the file. All others are skipped. (All other props are passed to physiol_cip_traceset)

Returns:

a_tss: Cell array of physiol_cip_traceset objects.

See also: `physiol_cip_traceset_fileset` (p. 152), `physiol_cip_traceset` (p. 147), `params_tests_dataset` (p. 113)

Author: Li, Su; Cengiz Gunay <cgunay@emory.edu>; and Jeremy Edgerton, 2007/12/18

A.50.27 Function `functions/string2File`

Summary: Writes string verbatim into a file.

Usage:

```
string2File(string, filename, props)
```

Parameters:

`string`**:** To be written into file.

`filename`**:** The file to be created.

`props`**:** A structure with any optional properties.

`append`**:** If 1, append to existing file.

Returns:

See also: cell2TeX (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/12/10

A.50.28 Function functions/sortedUniqueValues

Summary: Find unique rows in an already sorted matrix (or column vector).

Usage:

```
[rows, idx] = sortedUniqueValues(data)
```

Description: Uses the derivation by Matlab diff function method. Redundant with the Matlab function UNIQUE doing the same job: [rows, idx]= unique(data, 'rows', 'first'). However, sortedUniqueValues is more efficient if the input data is already sorted for some other reason (see usage in tests_db/invarValues).

Parameters:

`data`: A ascending row-sorted matrix or column vector.

Returns:

rows: A matrix or column vector of unique rows. idx: Indices of the unique rows in the original data matrix.

See also: uniqueValues (p. ??), unique (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/27

A.50.29 Function functions/meanSpikeFreq

Summary: Returns the mean firing frequency in Hz according to mean \ inter-spike-interval of the given spike train and the time resolution dt.

Usage:

```
meanFreq = meanSpikeFreq( spike_train, dt, period )
```

Description: Parameters: spike_train: Spike times returned by findspikes dt: Time step size [s] period: Duration of the total time period [dt]

See also:

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/03/08

A.50.30 Function functions/diff3T_h4

Summary: Estimate of third derivative using Taylor expansion (derived with same method as diffT and diff2T_h4).

Usage:

```
deriv2 = diff3T_h4(x, dy)
```

Description: d^3 x - x(k-2) + 2 * x(k-1) - 2 * x(k+1) + x(k+2) ——- = ———————————————————— dy^3 12 * dy^3

Parameters:

`x`: A vector of x = f(y).

`dy`: The resolution of the discrete points in the vector.

Returns:

deriv2: Estimate of the derivative.

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/04/18

A.50.31 Function functions/growRange

Summary: Returns the maximal range from rows of axis limits.

Usage:

```
range = growRange(ranges)
```

Parameters:

`ranges`: A matrix where each row is return val of axis func.

Returns:

range: The maximal range obtained that includes all given axes.

See also:

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/10/13

A.50.32 Function functions/readNeuronVecBin

Author: Konstantin Miller <miller@cs.tu-berlin.de>, Aug 09, 2005.

A.50.33 Function functions/colormapBlueCrossRed

Summary: Blue to red crossing colormap, with a black-colored zero-crossing.

Usage:

```
colors = colormapBlueCrossRed(num_half_colors)
```

Description: Colormap contains (2 * num_half_colors + 1) colors, where (num_half_colors + 1) is the zero crossing.

Parameters:

`num_half_colors`**:** Number of colors to generate on one of the red or blue scales.

`props`**:** A structure with any optional properties.

Returns:

colors: RGB color matrix to be passed to colormap function.

See also: `colormap` (p. **??**)

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/06/05

A.50.34 Function functions/diffT

Summary: Estimate of first derivative using Taylor expansion.

Usage:

```
deriv = diffT(x, dy)
```

Description: dx x(k-2) - 8 * x(k-1) + 8 * x(k+1) - x(k+2) —- = ———————————————— dy 12 * dy

Parameters:

`x`**:** A vector.

`dy`**:** The resolution of the discrete points in the vector.

Returns:

deriv: Estimate of the first derivative.

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/11/15

A.50.35 Function functions/interpValByIndex

Summary: Finds the interpolated value by using the real valued index from the data vector.

Usage:

```
val = interpValByIndex(idx, data)
```

Description: Parameters: idx: A real-valued index. data: A data vector.

Returns:

val: the value taken from the nearest integer indices of data and interpolated.

Example:

```
» a= [1 2 3];
» interpValByIndex(1.5, a)
ans =
1.5000
```

See also: spike_shape (p. 189)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/08/02

A.50.36 Function functions/subTextLabel

Summary: Draws a text label on a plot.

Usage:

```
handle = subTextLabel(x, y, text_str, props)
```

Parameters:

x, y: 2D coordinates.

text_str: String to be drawn on plot.

props: A structure with any optional properties.

Units: position units for the coordinates (see Units in axes properties).

Returns:

handle: Text object handle.

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/04/11

A.50.37 Function `functions/calcGraphNormPtsRatio`

Summary: Return the ratios of normalized to point units for dimensions of axis.

Usage:

```
[ratio_x, ratio_y] = calcGraphNormPtsRatio(grfx_handle)
```

Description: Used for findind character sizes given the size of an axis. Normally if the plot is resized, the characters may take up too much space or may not fit anymore unless the spacing is corrected.

Parameters:

`grfx_handle`**:** A graphics handle to an existing axis or figure.

Returns:

ratio_x, ratio_y: Normalized to points ratio for axis width and height, respectively.

Example:

```
To find the normalized distance for a 10pt character:
» dx = 10 * calcGraphNormPtsRatio(my_figure_handle);
```

Author: Cengiz Gunay <cgunay@emory.edu>, 2006/03/05

A.50.38 Function `functions/balanceInputProbs`

Summary: Balances samples according to prior class probabilities of the outputs.

Usage:

```
[new_inputs, new_outputs] = balanceInputProbs(a_class_inputs, a_class_outputs, balance_ratio,
props)
```

Description: Uses the method in Lawrence, burns, Back, Tsoi and Lee Giles "Neural network classification and prior class probabilities" for probabilitic balancing of input and output samples when the number of samples in each class is vastly different and causes problems with classification without balancing.

Parameters:

`a_class_inputs, a_class_outputs`**:** Input and output vectors.

`balance_ratio`**:** c_s, between 0 and 1. If 1, equal samples from each class if used. If 0, prior class probabilities are followed.

`props`**:** A structure with any optional properties.

`repeatSamples`**:** If 1, repeats the smaller class samples to match with larger class. Otherwise, takes the minimal number of samples to avoid repetitions (default=1).

Returns:

new_inputs, new_outputs: New input and output vectors.

See also: `approxMappingNNet` (p. **??**), `tests_db` (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2008/01/09

A.50.39 Function `functions/findspikes`

Author: Li, Su based on the original of Alfonso Delagado-Reyes

---

A.50.40 Function `functions/boxplotp`

Description: BOXPLOTP(X,NOTCH,SYM,VERT,WHIS,PROPS) produces a box and whisker plot for each column of X. The box has lines at the lower quartile, median, and upper quartile values. The whiskers are lines extending from each end of the box to show the extent of the rest of the data. Outliers are data with values beyond the ends of the whiskers.

A.50.41 Function `functions/collectspikes`

Author: <adelgado@biology.emory.edu>

A.50.42 Function `functions/findVectorInMatrix`

Summary: Finds rows of data that match row.

Usage:

```
idx = findVectorInMatrix(data, row)
```

Description: Matlab's eq (==) command unfortunately doesn't allow this directly.

Parameters:

`data`**:** A matrix or column vector.

`row`**:** A row vector.

`Returns`**:**

`idx`**:** Indices of matching rows in the original data matrix.

See also:

Author: Cengiz Gunay <cgunay@emory.edu>, 2005/09/1

A.50.43 Function functions/uniqueValues

Summary: Find unique rows in a matrix (or column vector).

Usage:

```
[rows, idx] = uniqueValues(data)
```

Description: Version which makes use of sort and diff. Maintains order of the original input.

Parameters:

data: A matrix or column vector

Returns:

rows: A matrix or column vector of unique rows. idx: Indices of the unique rows in the original data matrix.

See also: sortedUniqueValues (p. ??)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/24

A.50.44 Function functions/makeIdx

Summary: Prepare the idx structure from names.

Usage:

```
idx = makeIdx(names)
```

Description: Helper function.

Parameters:

names: Cell array of names for a db dimension.

Returns:

idx: Structure associating names to array indices.

See also: tests_db (p. 227)

Author: Cengiz Gunay <cgunay@emory.edu>, 2004/09/17

Function Index