**Welcome to Dragonsort beta**



Graphical user interface, chart, application

Description automatically generated

**Daniel Ko**

[**dsk13@ic.ac.uk**](mailto:dsk13@ic.ac.uk)

**Doc last updated: 28/02/20**

Contents

[Preface 3](#_Toc33802641)

[General workflow 5](#_Toc33802642)

[Main interface 6](#_Toc33802643)

[Manual manipulation 6](#_Toc33802644)

[Auto manipulation 7](#_Toc33802645)

[Scrub data 8](#_Toc33802646)

[Thresholding 8](#_Toc33802647)

[Data summary 8](#_Toc33802648)

[Data viewing 10](#_Toc33802649)

[Template generation 10](#_Toc33802650)

[Menu options 10](#_Toc33802651)

[Initialising 12](#_Toc33802652)

[Refine batch 13](#_Toc33802653)

# Preface

**A brutally short overview of spike sorting**

Spike sorting for extracellular signals generally has two approaches: **1) Feature-based clustering**. **2) Template matching**. While feature-based approach is more general, template matching tends to be faster and more accurate in some cases. The performance of feature-based clustering depends on the feature extraction algorithms (e.g. PCA, wavelet, neural network). To have sufficiently robust feature space, the sample size (spikes) must be large enough. The user would then have to decide on where to draw the boundaries between different clusters in the **feature space**. Templates, on the other hand, can be established with only a handful of spikes. If the spike shape is relatively constant, one can efficiently extract all the spikes that match the templates within given variation bounds. The user then must decide what is the acceptable variation in the **waveform shape**.

**Why Dragonsort?**

As an electrophysiologist, have you ever had the experience of seeing the “relevant spikes” not being picked up by a spike sorting software? You spent a few hours tweaking unintuitive parameters and the spikes still didn’t get included. Or, due to some artefacts, a few spikes of one spike train cannot be recognised by the spike sorting software but you need them for your analyses. The creation of the **Dragonsort** was based on the philosophy to give users **full control** over these decisions while making the process **transparent**. It first allows the user to construct the draft templates by selecting obvious spikes within the raw data. Then these templates are used to search within the data set with high matching criteria to confirm whether they are viable templates. The results are then **projected onto a feature space** for reference. The user then refine, merge, split templates to construct a reasonable set of templates. Dragonsort then allows the user to **relax the matching criteria** for the templates as the algorithm automatically assign raw spikes to the templates.

**What type of data are Dragonsort designed for?**

* Spikes with unconventional shapes that must be manually picked out to prime the sorter.
* Data with relatively small spike sample (i.e. < 500 spikes/unit) so direct feature-based statistics might not be powerful enough.
* Data with sparse spikes when losing spikes is critical to the interpretation of the data.
* Recordings with fewer than 8 channels.
* Recordings where you believe to contain fewer than 20 real single units.
* Recordings with outliers and artefacts that need to be removed manually.
* Data where well-isolated units are important and you do not want to deal with many multi-unit clusters.

**What gap does Dragonsort fill in the electrophysiology research?**

Electrophysiology recordings generally answers two types of questions: **1) activation level and circuit dynamics.** **2) neural encoding schemes**. **The first type of question** is typically done via extracellular recordings where a population of neurons can be monitored at the same time. The questions usually revolve around which neurons are active relative to some stimulus or reference neurons. Spike sorting for such investigation can afford to miss spikes under the assumption that a strict clustering criterion will only reduce the overall level of activity but not the relative activities. **The second type of question** is normally done via intracellular or patch recordings as we need to capture every single spike of a given neuron. This level of accuracy is needed to work out the biophysics and the information encoding scheme for the neuron of interest. However, this approach limits the number of neurons that can be recorded simultaneously simply due to the technical challenge (i.e. patching multiple cells at the same time). **Functional imaging** (e.g. calcium and voltage imaging) can easily replace the activation level questions but cannot easily replace electrophysiology for information encoding questions.

We are striking at the middle ground to answer the **information encoding questions while having access to a small population of neurons simultaneously**. Dragonsort was created to enhance the spike sorting accuracy to near-intracellular level on low channel count multi-electrode recordings. This will facilitate many areas such as multi-sensory integration and sensorimotor control. We encourage more researchers to join us in this middle ground to advance our understanding of neural circuits on the mechanistic level.

# General workflow

Below are the series of steps a typical spike sorting session will go through. In **bold** are the buttons/hotkeys you might use at each step.

1. [**Initialise**](#_Initialising) **(ctrl+I)** sorting by loading in a binary file OR **load (ctrl+L)** a previous sorting
   * Optional: **load templates (ctrl+K)** from a previous sorting
2. Auto-create junk units and candidate units
   * **Autocreate junk units** (identify noise spikes and remove them)
   * **Autocreate units** (some preliminary units are created based on PCA)
3. Curate and clean up the units created
   * **New unit (ctrl+N)** to add any units that should be there
   * **Split unit (ctrl+C)** if any MUA units seem easy to split
   * **Delete unit** for MUA situation for refine process to pick them up
   * **Toggle left unit as junk** to mark any noise units that we want to keep track of
   * **Merge unit (ctrl+V)** if any two units seem easy to merge
4. Populate units in batch – repeat until satisfactory number of units achieved
   * **Add spike (ctrl+D)**
   * **Force add (ctrl+G)**
5. Initial rough **refine batch (ctrl+X)** – assign ~50% of unassigned spikes
6. Make splitting/merging decisions on units –you can use **Undo (ctrl+Z)** to return to pre-split/merge state
   * If a unit looks like it might be formed of spikes from more than 1 cell, **split (ctrl+C)**
   * If two units look like they have very similar spikes, **merge (ctrl+V)**
7. Fine **refine batch (ctrl+X)**  – assign as many spikes as possible
8. **Remove spikes (ctrl+F)** to clean up units, split/merge units
9. **Next (ctrl+P)** batch
10. Optional: create more units manually or automatically (steps 2 and 3)
11. Repeat steps 4 - 9

# Main interface

Graphical user interface, chart, application

Description automatically generated

## Spike selection (both left unit figure and trace figure)

|  |  |  |
| --- | --- | --- |
| **Mode** | **Hotkey** | **Function** |
| Single selection | Left click on spike marker (in trace figure)/left click on spike trace (in left unit figure) | Selects/deselects a single spike for operation |
| Box selection | Left click on empty space | **First click** – places a marker to denote corner of selection box  **Second click** – places another marker to denote other corner of selection box then inverts selection of all spikes in the box |

## Manual operations

|  |  |  |
| --- | --- | --- |
| **Button** | **Hotkey** | **Function** |
| New unit | ctrl+N | Creates a new unit – use the coloured buttons at the bottom right or the dropdown above the unit figures to browse |
| Delete unit |  | Deletes unit currently viewed in the left unit figure |
| Add spike | ctrl+D | Brings up a popup copy of the data trace figure to select spikes in. These spikes will be run through the similarity check before being added to the current unit. |
| Force add spike | ctrl+G | Brings up a popup copy of the data trace figure to select spikes in. These spikes will be **NOT** be run through the similarity check before being added to the current unit. See add spike for hotkeys |
| Remove spike | ctrl+F | **Select spikes in the left unit figure to use this button**  Removes selected spikes from the current left unit |
| Remove plotted |  | Removes all currently plotted spikes from the current left unit |
| Split spike | ctrl+C | **Select spikes in the left unit figure to use this button**  Splits selected spikes from the current left unit into a new unit |
| Split plotted |  | Splits all currently plotted spikes from the current left unit into a new unit |
| Scrub through unit |  | Creates a popup that allows splitting and deleting spikes at specific points in time in the left unit |
| Merge units | ctrl+V | Merges left and right units into the right unit |

## Auto operations

|  |  |  |
| --- | --- | --- |
| **Button** | **Hotkey** | **Function** |
| Nice-fy unit numbers |  | Re-numbers units based on mean peak amplitude |
| Autosplit into: | ctrl+A | Uses PCA to split the current left unit into X units, where X is soft-determined by the number in the textbox to the right. This button will create a popup to check the results for confirmation or cancellation |
| Autosplit batch into: |  | Uses PCA to split the only the spikes in the current batch in the left unit into X units, where X is soft-determined by the number in the textbox to the right. This button will create a popup to check the results for confirmation or cancellation |
| Merge units | ctrl+V | Merges left and right units into the right unit |
| Toggle left unit as junk |  | Marks/unmarks the left unit as junk |
| Autocreate units |  | Automatically creates good units with peak amplitude BELOW a user-specified value (specified in a popup). PCA is used to cluster spikes, then a similarity check is used to further refine the clustering |
| Autocreate junk units |  | Automatically creates junk units with peak amplitude ABOVE a user-specified value (specified in a popup). PCA is used to cluster spikes, then a similarity check is used to further refine the clustering |

## Scrub data

|  |  |  |
| --- | --- | --- |
| **Button** | **Hotkey** | **Function** |
| Prev | ctrl+O | Load last batch |
| Next | ctrl+P | Load next batch |
| Go to batch |  | Go to batch X, where X is the number in the textbox to the right |
| Autosort to |  | Go to batch X, where X is the number in the textbox to the right. Sort each batch between the current batch and batch X by similarity check. |
| Autosort batch |  | Add/remove spikes from current batch to current existing units by similarity check. |
| Refine batch | ctrl+X | See section on [**Refine batch**](#_Refine_batch) |

## Menu options

|  |  |  |
| --- | --- | --- |
| **Button** | **Hotkey** | **Function** |
| Initialise | ctrl+I | See section on [**Initialising**](#_Initialising) |
| Load sorting | ctrl+L | Load a sorting |
| Load templates | ctrl+K | Load unit templates from another sorting. You can choose whether to import form the end of the dataset or the start |
| Save | ctrl+S | Save sorting (will bring up folder selection). Change savename on the top left textbox of main interface |
| Add time-stamp to save | ctrl+T | Automatically appends current timestamp to savenames on save |
| Undo | ctrl+Z | Undo/redo the last unit-altering action |
| Figure shortcuts – left unit figure |  | Ctrl+1 – activate cursor zoom, Ctrl+2 – reset zoom, Ctrl+3 – activate pan, Ctrl+4 – activate picker cursor |
| Figure shortcuts – data trace figure |  | Ctrl+Q – activate cursor zoom, Ctrl+W – reset zoom, Ctrl+E – activate pan, Ctrl+R – activate picker cursor |

## Thresholding

|  |  |
| --- | --- |
| **Button** | **Function** |
| Current units | Similarity threshold used in adding spikes to already existing units. Higher is more lenient – default 6 |
| New unit | Similarity threshold used in adding spikes to newly created units. Higher is more lenient – default 6 |
| Fuzzy thresholding | The *current units* and *new unit* thresholds will be used with a histogram analysis technique to optimise the threshold on a case-by-case basis |
| Spike detection | Any negative peaks below this value will be considered as a spike |

## Template generation

|  |  |
| --- | --- |
| **Button** | **Function** |
| Past batches | Use spikes in unit from X batches back to make similarity match templates |
| Future batches | Use spikes in unit from X batches in the future to make similarity match templates |
| # Spikes total | Total number of spikes to use in template generation for each unit |

## Data summary

|  |  |
| --- | --- |
| **Button** | **Function** |
| Show unit overview | Creates 2 popups. First popup shows all spikes in all/non-junk units, the second popup shows the PCA 3D plot of all spikes in all/non-junk units |
| Unit freq time series | Creates a popup that plots the number of spikes in each unit in each batch |
| Show loaded templates | **Only if unit templates were loaded in from other sortings**  Creates a popup that shows units against their initialisation templates |

## Data viewing

|  |  |
| --- | --- |
| **Button** | **Function** |
| Past batches | Plot spikes in unit from X batches back |
| Future batches | Plot spikes in unit from X batches in the future |
| # Spikes total | Total number of spikes to plot in the unit figures. Spikes will be plotted from youngest to oldest |
| Plot AllCh | **For multichannel data**  Enables plotting of data/units in their respective *AllCh* tabs. Performance may suffer |
| Batch size | Sample length of each batch |
| Trace y-limit | Sets the yaxis limits of the data trace figure |

# Initialising

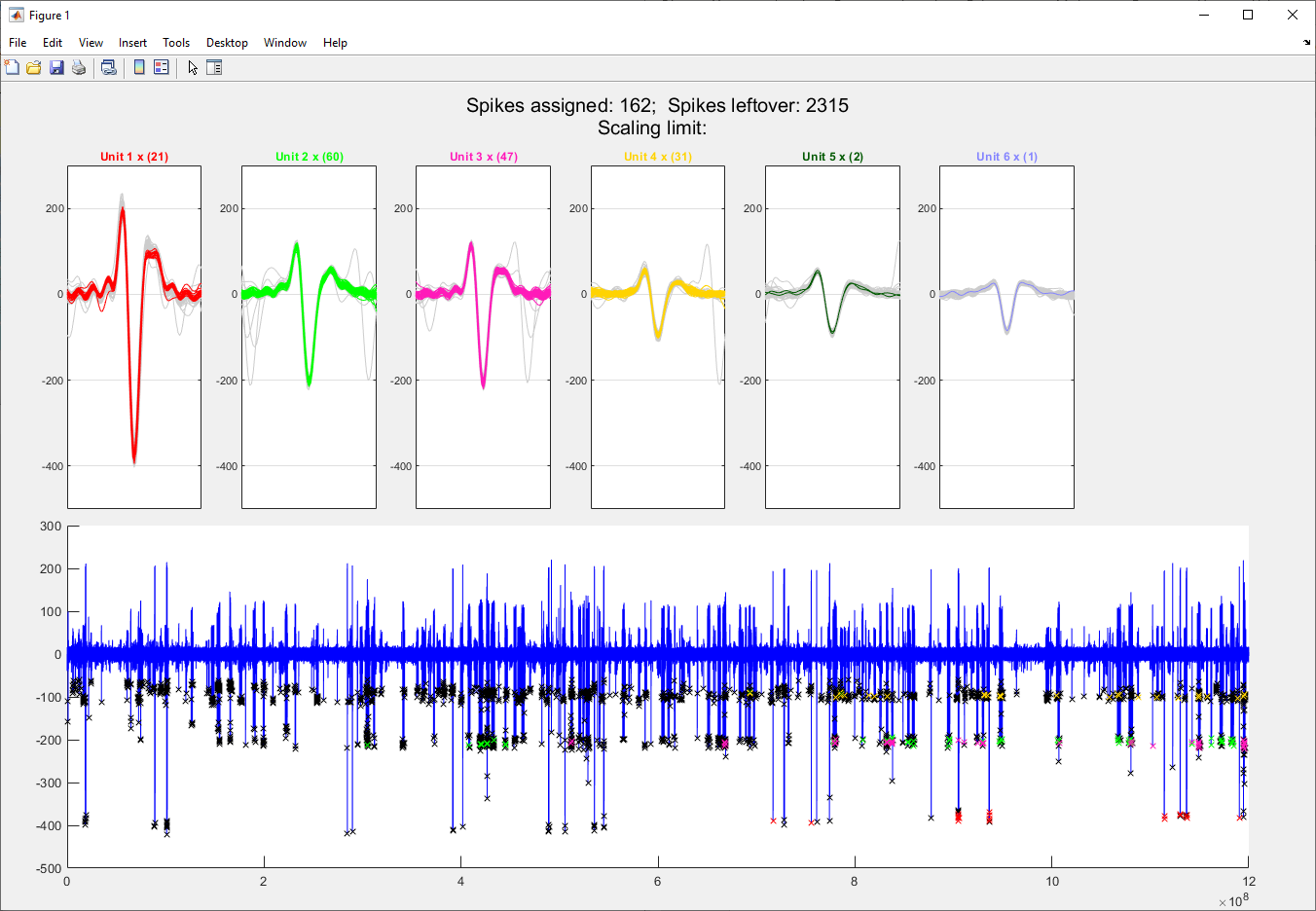
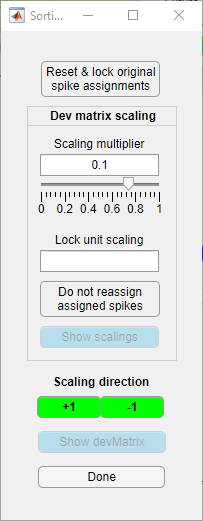
Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generated

1. **Session -> Initialise (or ctrl+I)**
2. Use the **Choose .bin** button to load in an experiment binary file. The binary file must be int16 and written column-wise.
3. In the case of interlaced data, tick the duplex data box instead of doubling the sampling rate in the box to the left.
4. Thresholds can be changed after initialisation, as can simple amplitude threshold which is used for initial spike detection.
5. The filtering options tab can be used to adjust the data pre-filtering.
6. Once initialised, you can load in unit templates from other sortings using **Session -> Load templates (ctrl+K)**, if desired.

# Refine batch



Refine batch works by scaling the similarity index of each detected spike to each cluster by a **Scaling multiplier**. If the similarity index of a spike for a unit drops below the similarity threshold for current units, the spike is assigned to that unit. The scaling of each unit can be individually locked if desired, to prevent unwanted spikes from entering the unit.

|  |  |
| --- | --- |
| Reset & lock pre-refine spike assignments | By default, the spikes in the current batch are reset and reassigned during the refining process. Toggling this button resets the current similarity index for all spikes for all units, then locks/unlocks original spike assignments for the current batch |
| Scaling lower limit | The lowest cumulative deviation scaling multiplier allowed – default 0.0001 |
| Scaling multiplier | Set global scaling multiplier for deviation indices |
| Scaling | Current scaling multiplier for the unit |
| ScalingLock | Tick this to prevent the ticked unit from being scaled |
| StealingLock | By default, spikes assigned to units in the refining process can be reassigned if the spike’s similarity index to another unit drops below the index to the current unit. Ticking this prevents this reassignment for the ticked unit |
| **Scaling direction** | Clicking +1 or -1 initiates a round of refinement, where the similarity indices of each spike are multiplied by the scaling multiplier in the direction of the button pressed |
| Done | Accept refinement and return to main interface |