Sagol School of Neuroscience
Tel Aviv University

Using NWB and DANDI for Neurophysiology Data

Contents

1. Introduction: Streamlining Neurophysiology Data

Neurophysiology research generates vast and complex datasets that are often difficult to share, reproduce, or analyze across labs due to inconsistent formats and metadata standards. **Neurodata Without Borders (NWB)**[1–4] addresses this challenge by offering a standardized, extensible data format designed to organize neurophysiology data and associated metadata in a structured and consistent way. This promotes interoperability, simplifies data analysis, and enables the development of reusable tools. Complementing this, is the **Distributed Archives for Neurophysiology Data Integration (DANDI)**, hosted by Amazon Web Services (AWS)[5]. DANDI serves as a cloud-based, free & open-access repository built specifically for NWB datasets. It not only facilitates data sharing and long-term storage but also enhances discoverability, supports collaboration, and provides computational tools for in-the-cloud data analysis through DANDI Hub. Both were created by the Brain Initiative[6,7].

By combining a common data standard with a powerful repository, NWB and DANDI create a robust ecosystem that empowers researchers to make their work more transparent, reusable, and impactful—accelerating progress in neuroscience.

Note 1: All info here is available online, mainly at https://docs.dandiarchive.org/, https://pynwb.readthedocs.io/en/stable/tutorials/index.html . The aim of this guide is to centralize it and provide a general overview. However, it is highly recommended to check the original documentations & guides.

Note 2: To interact with DANDI, some python knowledge is needed, through most interaction can be achieved via apps (GUIs) and the DANDI website. It is recommended to install python via conda distribution prior to working: https://docs.conda.io/projects/conda/en/latest/user-guide/install/index.html
And to create a conda environment using:

```
conda create --name DANDI_NWB
```

2. Understanding NWB: A Unified Format

NWB aims to be the universal data standard for neurophysiology, enabling data sharing, archiving, and the creation of common analysis tools. It tackles the problem of diverse data formats across labs, which makes comparing and replicating experiments difficult. NWB offers a comprehensive and extensible format that can handle raw and processed data from various neurophysiology techniques. Its key advantages include comprehensiveness, extensibility, usability, cross-platform compatibility, and support for tool development.

3. Practical NWB Usage: Structure and Tools

NWB files, with the .nwb extension, are based on the **Hierarchical Data Format (HDF5)**, which organizes data in a file-system-like structure. The fundamental building blocks are[8]:

- **Groups:** These are basically folders, capable of containing other groups and datasets. In these, a stage / type of data is usually found, as seen in the example below.
- **Datasets:** These are the containers for the actual data, typically n-dimensional arrays.

1. **Attributes:** Small pieces of metadata attached to groups or datasets, providing context. They may be very small and report the data type (for example `neurodata_type`).

- **Links:** References to other groups or datasets within the same NWB file, allowing for complex data relationships.

For example, At the top level, an NWB file typically contains several key groups[8,9]:

- **/general**: Stores metadata about the experiment, such as subject information, experimental setup, and lab details.
- **/acquisition**: Contains the raw data acquired during the experiment, which should

remain unchanged. This might include ElectricalSeries for electrophysiology data or ImageSeries for imaging data. This data should not be changed after placing.

- **/processing**: Holds processed data, typically the results of analysis pipelines. Data within this group can be modified. It often contains modality-specific sub-groups like /processing/ecephys for electrophysiology or /processing/ophys for optical physiology.
- **/analysis**: Reserved for lab-specific or custom analyses of the data.
- **/stimuli**: Stores information about the stimuli presented during the experiment.
- **/intervals**: Contains information about specific time intervals within the experiment, such as trials or epochs.

Each data object within the NWB file is often assigned a neurodata_type attribute, which specifies its semantic meaning within the NWB standard. For example, raw electrophysiology data is typically stored as an ElectricalSeries object, while spike times are stored as a SpikeTimes object. For a list of existing types in the base format, see the “basic neurodata types”.

Many parts of the NWB formats are built in object-oriented manner, where parts inherit from more global definitions – if a specific format does not match your needs, consider using a parent object (for example, a time series instead of ElectricalSeries). Also consider extensions to NWB – see the NWB Extension Catalog.

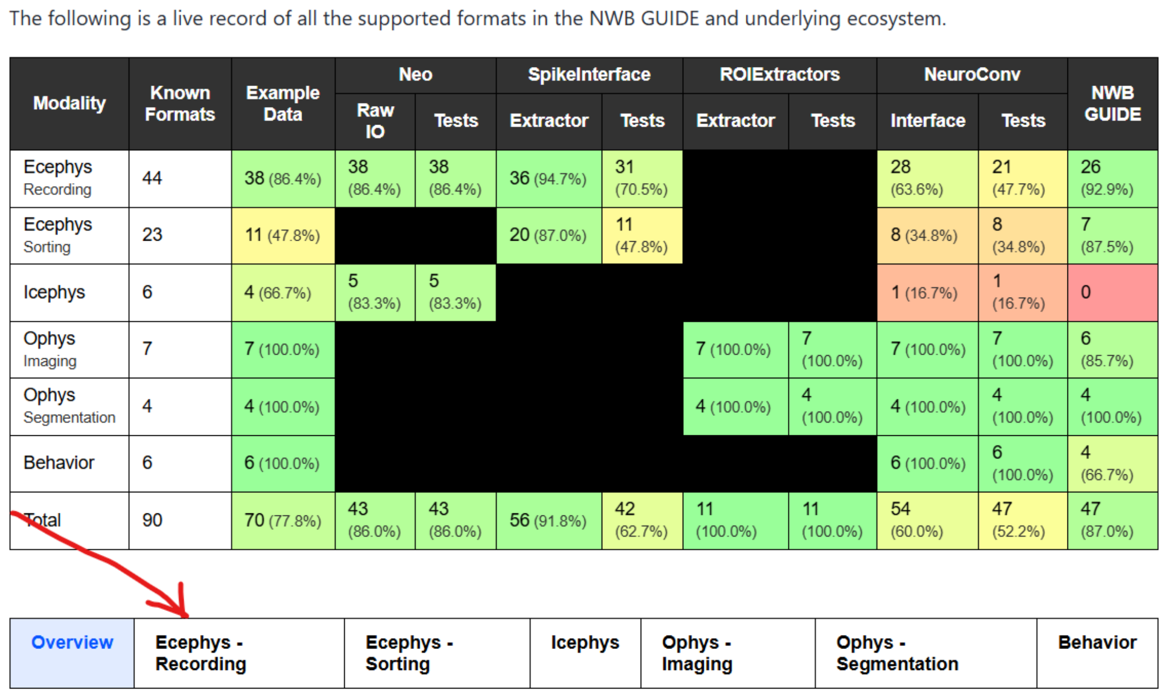
Finally, complex interactions with NWB files typically involves Python (with PyNWB[10]) or MATLAB (with MatNWB[11]). Note that to use PyNWB, you will need to install python – conda distribution is recommended (you can use Anaconda prompt as command line interface).

4. Converting Other data formats to NWB

See also: https://docs.dandiarchive.org/user-guide-sharing/converting-data/nwb/

There exist a few great tools for converting data into NWB format:

1. NWB GUIDE[12] – a GUI base program that can be used to import many extracellular recordings, calcium imaging and behavioural recordings to NWB. Installation is extremely simple, as it is a self-contained compiled program. To check if your data type is supported, use: https://nwb-guide.readthedocs.io/en/stable/format_support.html, click the type of data you want to import at the bottom:

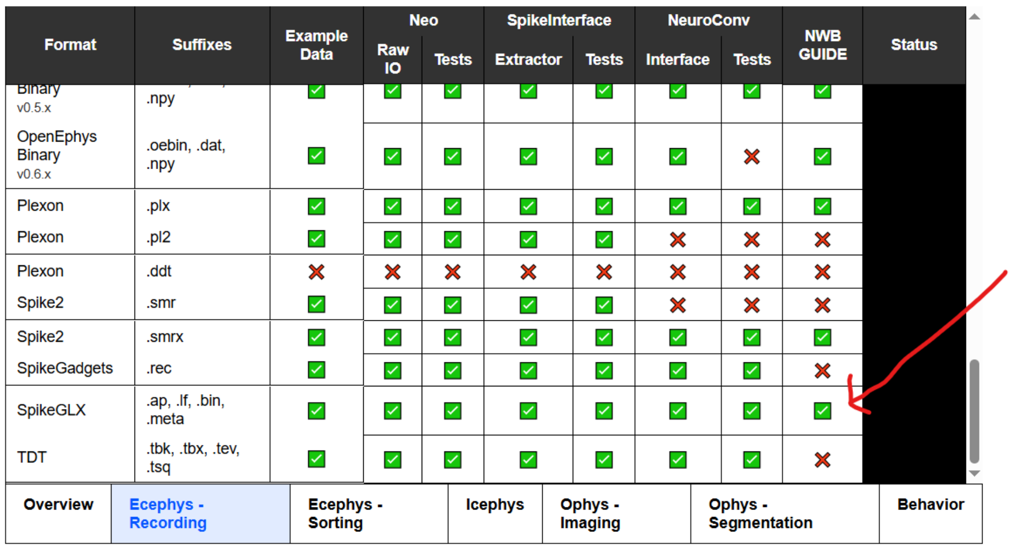


The following is a live record of all the supported formats in the NWB GUIDE and underlying ecosystem.

Modality	Known Formats	Example Data	Neo		SpikeInterface		ROIExtractors		NeuroConv		NWB GUIDE
			Raw IO	Tests	Extractor	Tests	Extractor	Tests	Interface	Tests	
Ecephys Recording	44	38 (86.4%)	38 (86.4%)	38 (86.4%)	36 (94.7%)	31 (70.5%)			28 (63.6%)	21 (47.7%)	26 (92.9%)
Ecephys Sorting	23	11 (47.8%)			20 (87.0%)	11 (47.8%)			8 (34.8%)	8 (34.8%)	7 (87.5%)
Icephys	6	4 (66.7%)	5 (83.3%)	5 (83.3%)					1 (16.7%)	1 (16.7%)	0
Ophys Imaging	7	7 (100.0%)					7 (100.0%)	7 (100.0%)	7 (100.0%)	7 (100.0%)	6 (85.7%)
Ophys Segmentation	4	4 (100.0%)					4 (100.0%)	4 (100.0%)	4 (100.0%)	4 (100.0%)	4 (100.0%)
Behavior	6	6 (100.0%)							6 (100.0%)	6 (100.0%)	4 (66.7%)
Total	90	70 (77.8%)	43 (86.0%)	43 (86.0%)	56 (91.8%)	42 (62.7%)	11 (100.0%)	11 (100.0%)	54 (60.0%)	47 (52.2%)	47 (87.0%)

Overview	Ecephys - Recording	Ecephys - Sorting	Icephys	Ophys - Imaging	Ophys - Segmentation	Behavior

And than check if there is a green V in the column named “NWB GUIDE”:



Format	Suffixes	Example Data	Neo		SpikeInterface		NeuroConv		NWB GUIDE	Status
			Raw IO	Tests	Extractor	Tests	Interface	Tests		
Binary v0.5.x	.npy	✓	✓	✓	✓	✓	✓	✓	✓	
OpenEphys Binary v0.6.x	.oebin, .dat, .npy	✓	✓	✓	✓	✓	✓	✗	✓	
Plexon	.plx	✓	✓	✓	✓	✓	✓	✓	✓	
Plexon	.pl2	✓	✓	✓	✓	✓	✗	✗	✗	
Plexon	.ddt	✗	✗	✗	✗	✗	✗	✗	✗	
Spike2	.smr	✓	✓	✓	✓	✓	✗	✗	✗	
Spike2	.smrx	✓	✓	✓	✓	✓	✓	✓	✓	
SpikeGadgets	.rec	✓	✓	✓	✓	✓	✓	✓	✗	
SpikeGLX	.ap, .lf, .bin, .meta	✓	✓	✓	✓	✓	✓	✓	✓	
TDT	.tbk, .tbx, .tev, .tsq	✓	✓	✓	✓	✓	✓	✓	✗	

Follow the GUI instructions and the tutorials in NWB GUIDE doc website to import your data.

2. NeuroConv[13]: A python based package for transforming everything NWB GUIDE can (extracellular recordings, calcium imaging and behavioural recordings) and supporting more formats and some intracellular recording. Check this list for supported formats: https://neuroconv.readthedocs.io/en/main/conversion_examples_gallery/index.html. While it requires some basic python knowledge, it is relatively easy to use and include full projects examples.
3. NWB Conversion Tools[14] – at the time of writing, still in alpha, but already being use by some labs. It also requires python knowledge & interface.

Lastly, if these tools do not help, you will have to write your own code using MatNWB or PyNWB to transform your data.

There are command line tools that are useful to **check that your NWB are well structured**:

1. pynwb-validate – included in your PyNWB distribution. Simply run: `pynwb-validate YOUR_NWB_FILE2CHECK.nwb`

This tool will only validate the file is matching NWB schema, and therefore OK and can be uploaded to DANDI. It should be possible to use pynwb-validate of files created using MatNWB. So far, there is no validator integrated into MatNWB.

2. NWB Inspector[15] – needed to install in your python environment. Afterward, run: `nwbinspector path/to/my/data/dir/`
   This tool will check your entire data directory. Instead of validating general schema, it will try to validate your data make sense (time & data matches, etc) and adherence to the strictest best practice recommendations. It should work with data created in MatNWB as well.

5. Requirements and Limitations of NWB and DANDI

Using NWB programmatically requires proficiency in Python or MATLAB (version compatibility to be checked) and installation of the respective NWB library (PyNWB or MatNWB). NWB files are based on the HDF5 format. While NWB is versatile, the internal folder structure can vary. Metadata quality in DANDI can differ across datasets. DANDI require a DANDI account, **and data uploaded to DANDI is usually public, including "drafts"!** Note that when you think about uploading your data – you shouldn't upload something you cant share as a test. Large datasets on DANDI might require streaming due to download limitations. Converting existing data to NWB can be challenging. NWB adoption is growing but not yet universal.

6. Exploring DANDI: A Data Repository

DANDI[16,17] is a public repository for neurophysiology datasets, primarily in NWB format. It allows researchers to share, publish, and process their data in a standardized way. DANDI validates uploaded NWB files and extracts metadata for improved discoverability. It also provides programmatic access via an API and offers computational resources through DANDI Hub. Using DANDI facilitates data discovery, sharing, collaboration, and access to computational resources. The archive hosts a growing collection of diverse neurophysiology datasets.

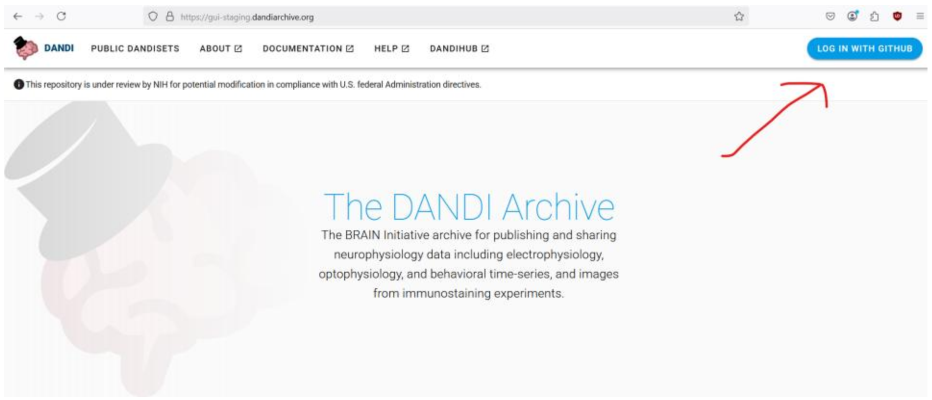
7. Uploading Data to DANDI: Steps, Cost, and Space

The central repository for DANDI is: https://dandiarchive.org/ .

For "staging", development, learning and testing, use: https://gui-staging.dandiarchive.org/ .

Before upload, you must create an account (which takes up to 2 business days!):

Register to DANDI using your GitHub account:



This repository is under review by NIH for potential modification in compliance with U.S. federal Administration directives.

The DANDI Archive

The BRAIN Initiative archive for publishing and sharing neurophysiology data including electrophysiology, optophysiology, and behavioral time-series, and images from immunostaining experiments.

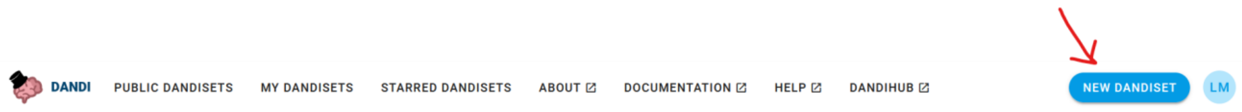
You will need to fill in some details about you and how you are planning to use DANDI. Wait for approval – it can take couple of days. Note that the "development" & "main" servers are treated as a separated users, and you will need to register to each of them separately, even if using the same github account.

An account connected to your tau mail is recommended for faster approval.

To upload your data, you can:

1. **Option 1 for "simple" datasets: use GUI:** you can use NWB GUIDE to upload your data. This work especially well if you already used it to create your NWB dataset.
2. **Option 2 for "complex" datasets: use website & command line:**

   A. **Create a Dandiset:** Create a new "Dandiset" using the button next to your login:



A "Dandiset" is a collection of multiple NWB datasets, which usually represent a

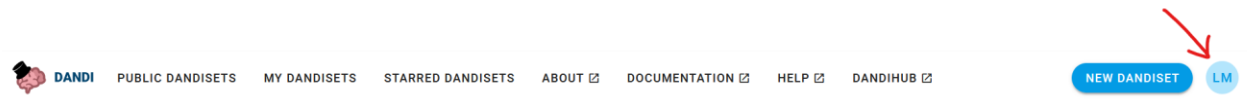
single experiment.
Note that Dandisets are always public, even as drafts, unless you have an approval from an NIH grant. In that case, data is not completely public, but metadata (such as experimental design) is.
For non embargoed Dandisets, you must choose either "*Creative Commons Zero v1.0 Universal*" (spdx:CC0-1.0), or "*Creative Commons Attribution 4.0 International*" (spdx:CC-BY-4.0), the main difference to my understanding is that CC-BY-4.0 require users to give you attribution.

B. **Convert Data to NWB:** Ensure your data is in the NWB format and validated using pynwb-validate, and does not return any critical issues using NWB Inspector. DANDI will also check this, but it is better to make sure everything OK before performing a possibly lengthy upload of a large dataset. Consider also using.

C. **Update Metadata:** Make sure your data have an NWB file containing Subject object with metadata for each subject. Subject object must contain subject_id, species (Latin or NCBI), sex & date_of_birth or age attribute, in ISO 8601 format. For more info, check:
https://pynwb.readthedocs.io/en/stable/tutorials/general/plot_file.html#subject-information
https://docs.dandiarchive.org/user-guide-sharing/validating-files/#missing-dandi-metadata

D. **Install DANDI Client:** Install the DANDI command-line interface (CLI) using pip: `pip install -U dandi` or `conda install -c conda-forge dandi` if you are using conda.

E. **Obtain API Key:** Get your API key from your DANDI archive profile on the website. Just click the small circle with your user name, and it will be under there:



After coping it, set it as an environmental variable in the same command line you are using DANDI, using:

`export DANDI_API_KEY=personal-key-value` in mac/Linux & `export DANDI_API_KEY=personal-key-value` in windows.

F. **Follow the steps in:** https://docs.dandiarchive.org/user-guide-sharing/uploading-data/ . This is basically copying the dandiest locally, organize the data in it, validate that everything is OK and upload.

Cost and Storage Space:

- **Cost:** The DANDI archive provides **free storage and access** to neurophysiology data, supported by the Amazon Web Services (AWS) Open Data Sponsorship Program. There is no subscription fee required to upload or download data. The project is funded by grants, primarily from the NIH.
- **Storage Space:** There is a **limit of 5TB per file** that can be uploaded to DANDI. While there isn't a strict limit on the total size of a Dandiset, **if you plan to upload more than 10TB of data, it is recommended to contact the DANDI team**. The archive currently holds a significant amount of data (hundreds of terabytes) and has sufficient capacity for the moment. Versioned items are retained for the lifetime of the NIH award, which currently expires in April 2029.

By following these steps, you can effectively convert your neurophysiology data to the standardized NWB format and share it with the broader scientific community through the DANDI archive.

8. More Resources

1. NWB Workshops and Hackathons: https://neurodatawithoutborders.github.io/nwb_hackathons/.
2. List of all brain imitative tools: https://www.braininitiative.org/toolmakers-resources/.
3. NWB & DANDI schema: https://www.nwb.org/nwb-software/
4. DANDI helpdesk: https://github.com/dandi/helpdesk/issues/new/choose

5. Video DANDI tutorial (2021): https://www.youtube.com/watch?v=fFnx-wzlLOs

9. Domain specific areas:

This part of the guide is intended provide you with an overview on how to add specific datatype using coding.

For official examples on how to use PyNWB to create your NWB files, check out: https://pynwb.readthedocs.io/en/stable/tutorials/index.html#domain-specific-tutorials

And here for MatNWB:
https://nwb.org/matnwb/#tutorials

Additionally, for most systems the raw data and streamlined common analysis can be changed to NWB using things in 5. Converting Other data formats to NWB, leaving you only needing to convert the unique processing you did for your experiment.

9.1. Extracellular Electrophysiology

See also:
https://pynwb.readthedocs.io/en/stable/tutorials/domain/ecephys.html#sphx-glr-tutorials-domain-ecephys-py

As always, you start by creating an "NWBFile" object, with all the metadata, then you start adding objects into it. After that, we can focus on adding the electrophysiological data:

First create a "Device" for the entire electrode, using "create_device" method of the NWB file. Then create "electrode_group" using the "create_electrode_group" method of the NWB file for each shank and use it as an argument to the "add_electrode" method of the file to pass each recording site (channel).

Now to add the raw data -using "create_electrode_table_region" you can recollect a subset of electrodes.

Use a specific row in that table to create an "ElectricalSeries" - this is an object that is specialized to hold time series of voltage data. Use it to hold your raw data, in reference to a specific electrode group. Moving along the first dimension ("rows") move through different time points, while moving along the second dimension ("columns") move through different channels.

Spike detections are also added directly into the nwbfile using the "add_unit_column" method. After adding this column, simply use "addunit" to specify the spike times of a whole unit. Note that this is true only for sorted spikes. Unsorted spikes / multiunits are stored separately, see official linked guide above for more info.

For processed data, you need to add a processing module into the NWB file, using "create_processing_module". After adding the module, you can add processed data using the module "add" method.

For example, LFP is created using an "ElectricalSeries" as above, where the electrodes property say to us which electrodes groups were used to create that LFP. The "ElectricalSeries" is passed to an "LFP" object, which is than added to the processing module as above.

When finished, don't forget to write your NWB file – in PyNWB, it is done using the context manager "NWBHDF5IO", see official guide for more info.

9.2 Calcium Imaging

See also: https://pynwb.readthedocs.io/en/stable/tutorials/domain/ophys.html.

As always, you start by creating an "NWBFile" object, with all the metadata, then you start adding objects into it.

First create a "Device" for the microscope, using the "create_device" method of the NWB file, and an "OpticalChannel" explaining what emission wavelength was used.

Those two obejcts are used in the “create_imaging_plane” method of the NWB file, creating an “ImagingPlane” object that holds information about what area was recorded & how.

Now it is time to add the raw data: create either “OnePhotonSeries” or “TwoPhotonSeries” – both hold the data in the “data” property, as a 3D or 4D array where the first dimension is frame number and the others are locations (x, y & z). Note that they both are needing the “ImagingPlane” as input. After creating this data holder, add it to the data using the “add_acquisition” method of the NWB file.

Now we move to the processing: using the “create_processing_module” method of the NWB file, create a “ProcessingModule” to add the processed info into. You can add information into it by using the “ProcessingModule” “add” method. To add the ROIs, create an “ImageSegmentation” object, and use its “create_plane_segmentation” to create an “PlaneSegmentation” dynamic table object. Add it to the “ProcessingModule”. Using the “add_roi” of this “PlaneSegmentation” object, you can add a 2-dimensional mask or a list of coordinates to specify the ROIs in the original image.

To add fluorescence measurements of ROIs in this “PlaneSegmentation”, use its “create_roi_table_region” method to collect specific ROIs, using the “region” property and referencing them by index. Next, using the selected ROIs, create an “RoiResponseSeries” to hold the fluorescence data – it’s data first dimension is frames, the second is ROI number. Pass the “RoiResponseSeries” to a “Fluorescence” or “DfOverF” object, and than add it to the “ProcessingModule” using the “add” method, as mentioned above.

When finished, don’t forget to write your NWB file – in PyNWB, it is done using the context manager “NWBHDF5IO”, see official guide for more info.

9.3 Intracellular Electrophysiology

See Also https://pynwb.readthedocs.io/en/stable/tutorials/domain/plot_icephys.html.

First, consider using NeuroConv - it support AXON .abf files conversion directly.

If you do need to use PyNWB or MatNWB to create the NWB files "Manually", you will have to extract the data from your recording software to your program language. For example, if you are using .abf files (Axon Clampex / Clampfit / pCLAMP[18]) consider using **abfload**[19] for MATLAB and **pyABF**[20] for python.

As always, you start by creating an "NWBFile" object, with all the metadata, then you start adding objects into it.

Use the "create_device" method of the file object to create a device (what equipment you used for the recording), and the "create_icephys_electrode" method of the file to create an object for the recording electrode.

Next you will need to create your stimulus – response pairs. For each sweep, use either "VoltageClampStimulusSeries" or "CurrentClampStimulusSeries" to describe your injected current/voltage. Use either "VoltageClampSeries", "CurrentClampSeries" or "IZeroClampSeries" (when current is clamped to 0) to describe your recorded responses.
After creating a pair, use the "add_intracellular_recording" method of the NWB file to add them.

This is basically it – see the official guide linked above for more options, such as adding custom columns with information about each sweep, adding simultaneous & sequential recording, adding repetitions and adding experimental conditions.

When finished, don't forget to write your NWB file – in PyNWB, it is done using the context manager "NWBHDF5IO", see official guide for more info.

Bibliography

1. The NWB Data Standard – Neurodata Without Borders. https://www.nwb.org/nwb-neurophysiology/.
2. Neurodata Without Borders – The Kavli Foundation. https://www.nwb.org/.
3. Rübel, O. *et al.* NWB:N 2.0: An Accessible Data Standard for Neurophysiology. 523035 Preprint at https://doi.org/10.1101/523035 (2019).
4. Teeters, J. L. *et al.* Neurodata Without Borders: Creating a Common Data Format for Neurophysiology. *Neuron* **88**, 629–634 (2015).
5. Rübel, O. *et al.* The Neurodata Without Borders ecosystem for neurophysiological data science. *eLife* **11**, e78362 (2022).
6. Home | BRAIN Initiative. https://braininitiative.nih.gov/.
7. The BRAIN Initiative Alliance. *The BRAIN Initiative Alliance* https://www.braininitiative.org/.
8. Welcome to the NWB Format Specification — NWB Format Specification 2.9.0-alpha documentation. https://nwb-schema.readthedocs.io/en/latest/index.html.
9. Anatomy of an NWB File — NWB Overview 0.1 documentation. https://nwb-overview.readthedocs.io/en/latest/intro_to_nwb/2_file_structure.html.
10. NWB for Python — PyNWB 2.8.3 documentation. https://pynwb.readthedocs.io/en/latest/.
11. matnwb. *matnwb* https://neurodatawithoutborders.github.io/matnwb/.
12. NWB GUIDE documentation — NWB GUIDE 1.0.5 documentation. https://nwb-guide.readthedocs.io/en/stable/index.html.
13. NeuroConv — NeuroConv documentation. https://neuroconv.readthedocs.io/en/main/.
14. Welcome to the documentation for NWB Conversion Tools! — NWB Conversion Tools documentation. https://nwb-conversion-tools.readthedocs.io/en/main/index.html.
15. NWB Inspector — NWBInspector documentation. https://nwbinspector.readthedocs.io/en/dev/.
16. DANDI. *DANDI* https://about.dandiarchive.org/.

17. DANDI Archive. https://dandiarchive.org/.
18. Axon^{TM}pCLAMP^{TM} 10 Electrophysiology Data Acquisition & Analysis Software Download Page. https://support.moleculardevices.com/s/article/Axon-pCLAMP-10-Electrophysiology-Data-Acquisition-Analysis-Software-Download-Page.
19. GitHub - fcollman/abfload: matlab fuction for reading abf files including v2.0 files. https://github.com/fcollman/abfload.
20. pyABF - A Simple Python Library for Working with ABF Files. https://swharden.com/pyabf/.