Tucker-Davis Technologies
Frontiers in Neuroscience

TDT Matlab Tools

Introduction

TDT provides a set of tools for interacting with TDT data in Matlab. These tools install into the directory C:\TDT\OpenEx\Examples\TTankX_Example\.

The latest version can be downloaded here:
http://www.tdt.com/files/examples/OpenDeveloperExamples.zip

To begin, set Matlab's current folder to the location of TDT2mat.m, or add the location to your Matlab path with the addpath command:

```
>> addpath('C:\TDT\OpenEx\Examples\TTankX_Example');
```

TDT Data Storage

Data collected or used by TDT software is stored in tanks—special directories on your hard drive. Each time you press 'Record' in Synapse, a new block is created within the tank. Within a block different stores can record different types of events at different rates. The blocks are special folders within the tank directories.

TDT2mat

TDT2mat is an all-in-one function for reading TDT data into Matlab. It needs only one input, the block path.

```
>> data = TDT2mat('C:\TDT\Synapse\Tanks\Exp1-160921-120606\Subject1-160921-124036');
```

TDT2mat will return a structure containing all recorded data from that block, organized by type:

1. **epocs** are values stored with onset and offset timestamps that can be used to create time-based filters on your data. They can be created by Epoch Data Storage gizmos and stimulation gizmos, among others.
2. **streams** are continuous single channel or multichannel recordings, like those stored by Stream Data Storage gizmo. The structure includes the data array and sampling rate.
3. **snips** are short snippets of data collected on a trigger. For example, action potentials recorded around threshold crossings in the PCA Spike Sorting gizmo, or fixed duration snippets recorded by the Strobe Store gizmo. This structure includes the waveforms, channel numbers, sort codes, trigger timestamps, and sampling rate.
4. **scalars** are similar to epocs but can be single or multi-channel values and only store an onset timestamp when triggered. These can be created by the Strobe Data Storage gizmo.

TDT2mat uses parameter value combinations to refine the imported data. To extract specific event types only, use the 'TYPE' option. For example, to import epocs and snippets only, use this:

```
>> data = TDT2mat('C:\TDT\OpenEx\Tanks\EXAMPLE\Block-1','TYPE',{'epocs','snips'})
```

Use the 'STORE' option to extract a particular data store by name, in this example a streaming event called PDec. Combine this with the 'CHANNEL' option to extract a single channel, in this case channel 2:

```
>> data = TDT2mat('C:\TDT\OpenEx\Tanks\EXAMPLE\Block-1','STORE','PDec','CHANNEL',2);
```

You can also filter by time, if you are only interested in portions of the recording, or if the entire recording won't fit into available memory (RAM) at one time. Use the 'T1' and 'T2' options to specify the start and stop time, in seconds, to retrieve from the block. This example reads only the second minute of the block into Matlab:

```
>> data = TDT2mat('C:\TDT\OpenEx\Tanks\EXAMPLE\Block-1', 'T1', 60, 'T2', 120);
```

TDT2mat offers many more useful options that are described in its help section.

```
>> help TDT2mat
```

In addition, it is designed to be used even when recording is taking place. This allows analysis to take place on data as it is stored into the data tank. For more information, see the SynapseLive demo in the Synapse API.

TDTfilter

TDTfilter applies advanced epoc filtering to extracted data. For example, if you only want to look at data around a certain epoc event, you will use TDTfilter to do this. See Raster_PSTH.m included in the examples zip file for a complete demonstration.

To only look at data around the epoc event timestamps, use a 'TIME' filter. In this example, data from 20ms before the *Levl* epoc to 50ms after the onset is retained.

```
>> data = TDTfilter(data, 'Levl', 'TIME', [-0.02, 0.07]);
```

To only look at data when an epoc was a certain value, use a 'VALUE' filter. In this example, only data when the *Freq* epoc was equal to 9000 or 10000 is retained.

```
>> data = TDTfilter(data, 'Freq', 'VALUES', [9000, 10000]);
```

If you want to look for a particular behavioral response that occurs sometime during the allowed time range, use the 'MODIFIERS' filter. In this example, only data when the *Freq* epoc was 10000 AND the *Resp* epoc had a value of 1 sometime during the *Freq* epocis retained.

```
>> data = TDTfilter(data, 'Freq', 'VALUES', [10000]);
>> data = TDTfilter(data, 'Resp', 'MODIFIERS', [1]);
```

As you can see, for complex filtering the output from one call to TDTfilter can become the input to the next call to TDTfilter. If your data sets are large, or if you are iterating through many combinations of epoc variables, it is preferred to extract only the epocs and do all of the epoc filtering first to find the valid time ranges that match the filter, and then use this as the ‘RANGE’ input to TDT2mat to extract all events (including snips, streams) on only those valid time ranges.

```
>> data = TDT2mat(block_path, 'TYPE', {'epocs'});
>> data = TDTfilter(data, 'Freq', 'VALUES', [9000, 10000]);
>> data = TDTfilter(data, 'Levl', 'VALUES', [70, 80, 90]);
>> data = TDTfilter(data, 'Resp', 'MODIFIERS', [1]);
>> data = TDT2mat(block_path, 'RANGE', data.time_ranges);
```

TDTdigitalfilter

TDTdigitalfilter mimics the digital filters that run on TDT hardware in real-time. If you record raw unfiltered data you can pass it through TDTdigitalfilter for post-processing.

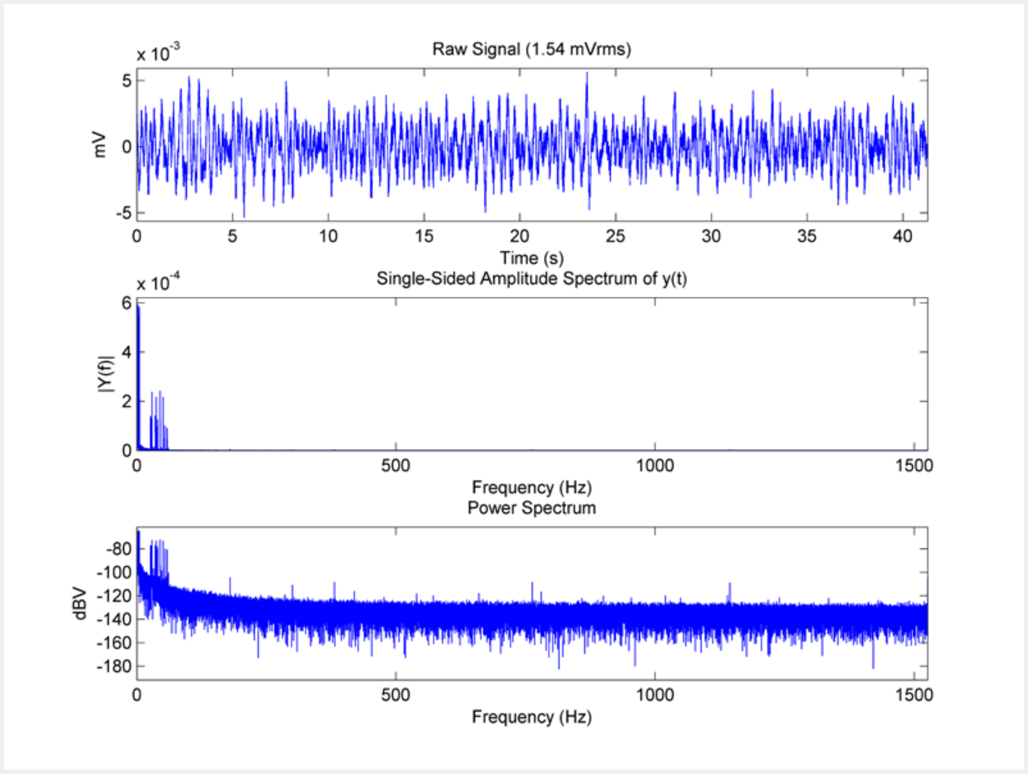
```
data = TDT2mat(block_path, 'VERBOSE', 0); % 'VERBOSE' option can disable text output
data.streams.Raw1 = TDTdigitalfilter(data.streams.Raw1, [300, 5000], 'SU');
```

You may recognize slight delays in the filtered signal, since the hardware filtering are not zero phase filters. The amount of delay depends on the hardware system rate (how fast the DSPs are running) and filter order.

TDTfft

TDTfft performs frequency analysis on stream data extracted by TDT2mat. You only need to provide the variable containing your data and the channel of interest to obtain full spectral information from the collected data.

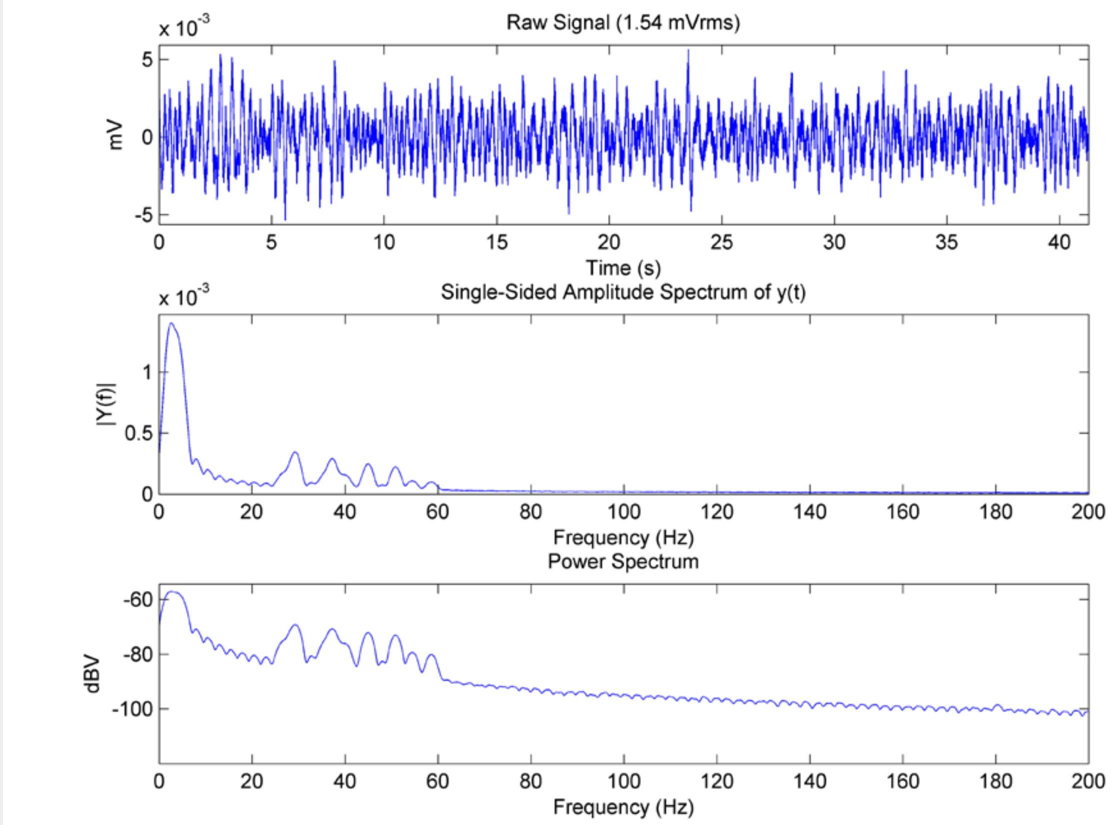
```
data = TDT2mat(block_path);
CHANNEL = 1;
TDTfft(data.streams.Wav1,CHANNEL);
```



TDTfft displays a figure containing three plots: 1) raw signal, 2) spectral amplitude, and 3) spectral power. By default, the frequency plots are all calculated with one-sided fast Fourier Transform (FFT), so the upper frequency limit is shown as half of the sampling rate of the streaming data (Wav1 in this case).

In this example, most of the spectral power lies within the first 200Hz. To extract only the first 200Hz signal and obtain higher resolution, we can use the 'FREQ' option. Also, we can split the segment into multiple smaller segments and average the spectrums of the shorter segments together to smooth out noise seen in power spectrum. The 'NUMAVG' option is the number of windows used for smoothing.

```
TDTfft(data.streams.Wav1,CHANNEL,'NUMAVG',100,'FREQ',[0 200]);
```



To also see a spectrogram of how the power spectrum changes over time, set the 'SPECPLOT' option to 1.

```
TDTfft(data.streams.Wav1,CHANNEL,'NUMAVG',100,'FREQ',[0 200],'SPECPLOT',1);
```