Real-Time Neural Spike Detection

Neuralink, US 2020/041372, Filed 9/9/2020, Published 1/21/2021

5/30/2024

Background

- **Key information** of neural activity can be identified based on **neuron spikes** - action potential in neuron
- **Spikes** - **characteristic rises** in **voltage** across **plasma membrane** in cell, caused by **depolarization + repolarization** in membrane
- Extant methods:
  - Total voltage exceeding some threshold - lots of false positives
  - Machine learning - very expensive, great deal of power

How to Detect Neuron Spikes in Real Time

- Purpose: To detect and classify characteristic signals such as brain neuron spikes
- Table of contents:
  - What is a spike?
  - Integrated Chip Design
  - Compression Engine
  - Basics: Floating vs Fixed-Point Computation
  - Overall Procedure:
    1. Receive Biological Signal
    2. Filter Signal
    3. Smooth Signal
    4. Identify Fit Values
    5. Compute Charateristic Values
    6. Determine Threshold Values
    7. Determine whether spike(s) are detected
    8. Transmit Indication of Spike
    9. Additional: Classify Spikes

What is a spike?

<u>Spike</u> - characteristic rises & falls

- Start out with **initial resting value**
- **First positive change** in amplitude
- **Reduction** in amplitude **below initial resting value**
- **Second positive change** in amplitude **greater than first positive change**

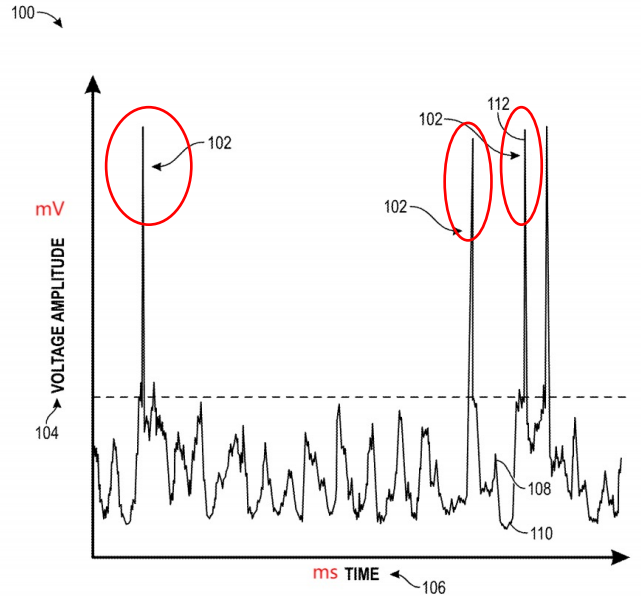
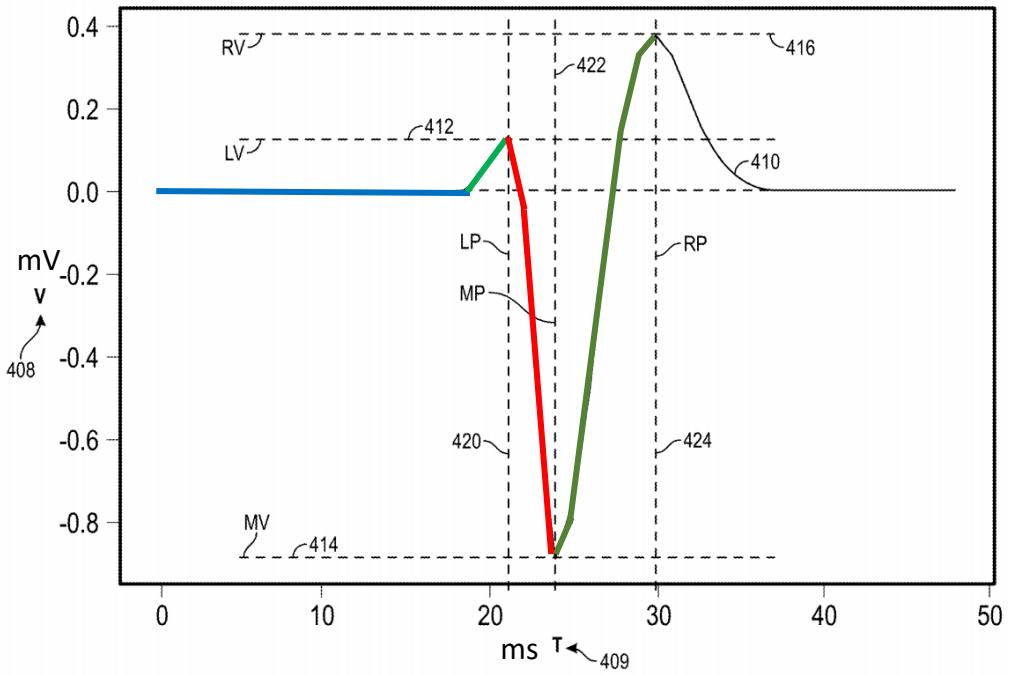


FIG. 1



Integrated Chip Design

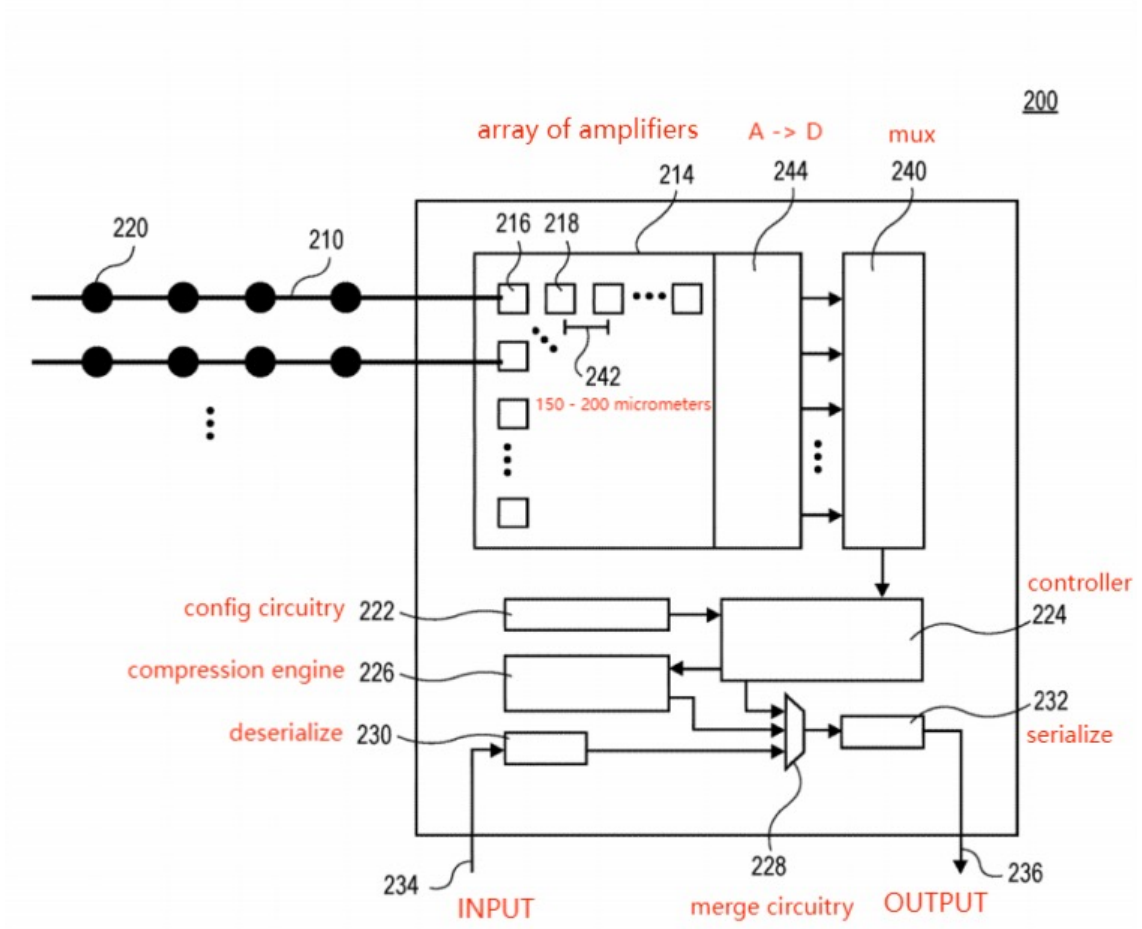


FIG. 2

Compression Engine

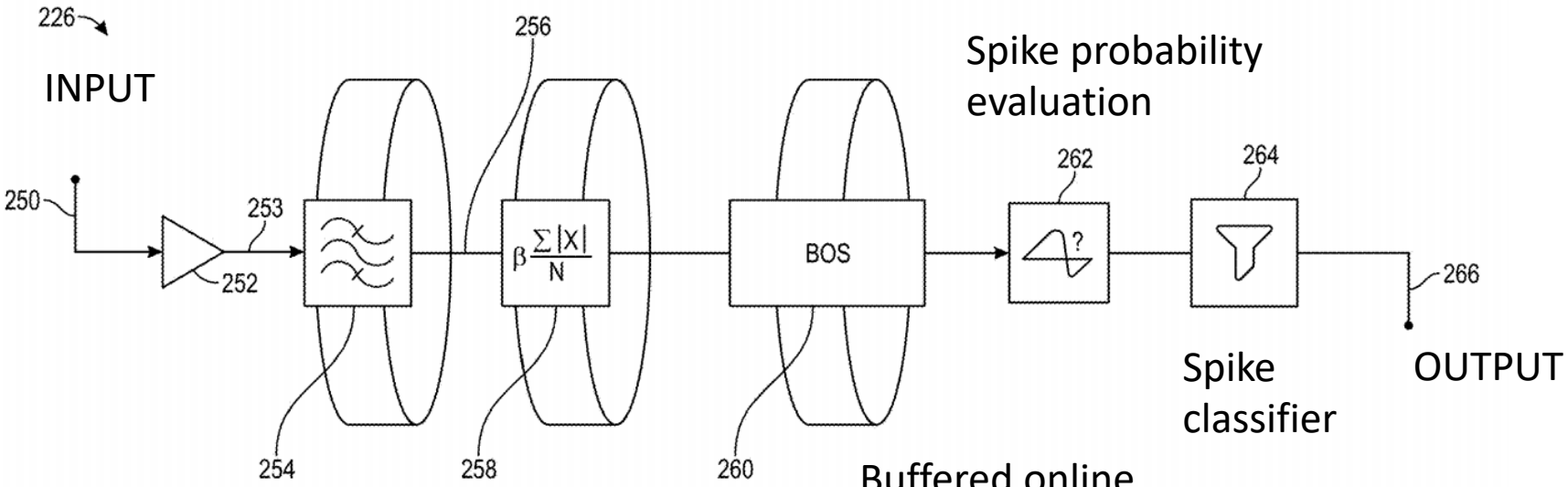


FIG. 2B

Basics: Floating Point Representations

- **ANY** number of decimals
- **Conventional** - **Very inefficient** & **resource intensive** computationally
- Used in **software** implementations (Python)

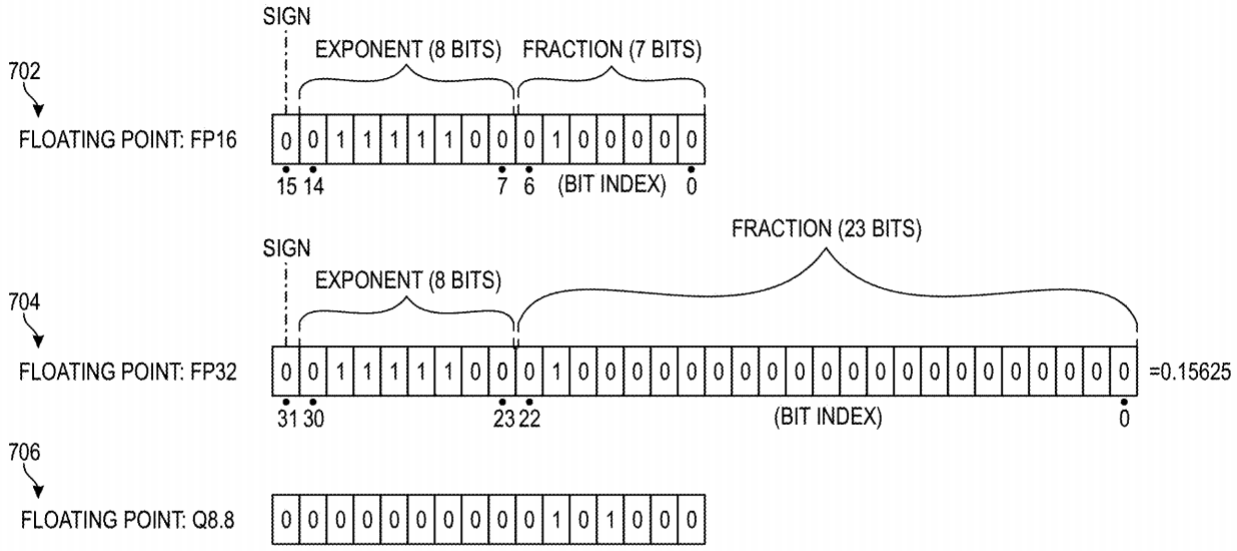


FIG. 7

Basics: Fixed Point Representation

- **FIXED** number of decimals
- More **computationally efficient**
- ex. **Q7.14**: allocate **7 bits for number (exponent)**, **14 bits for fraction**
- Used in **hardware** implementations (rarely in software)

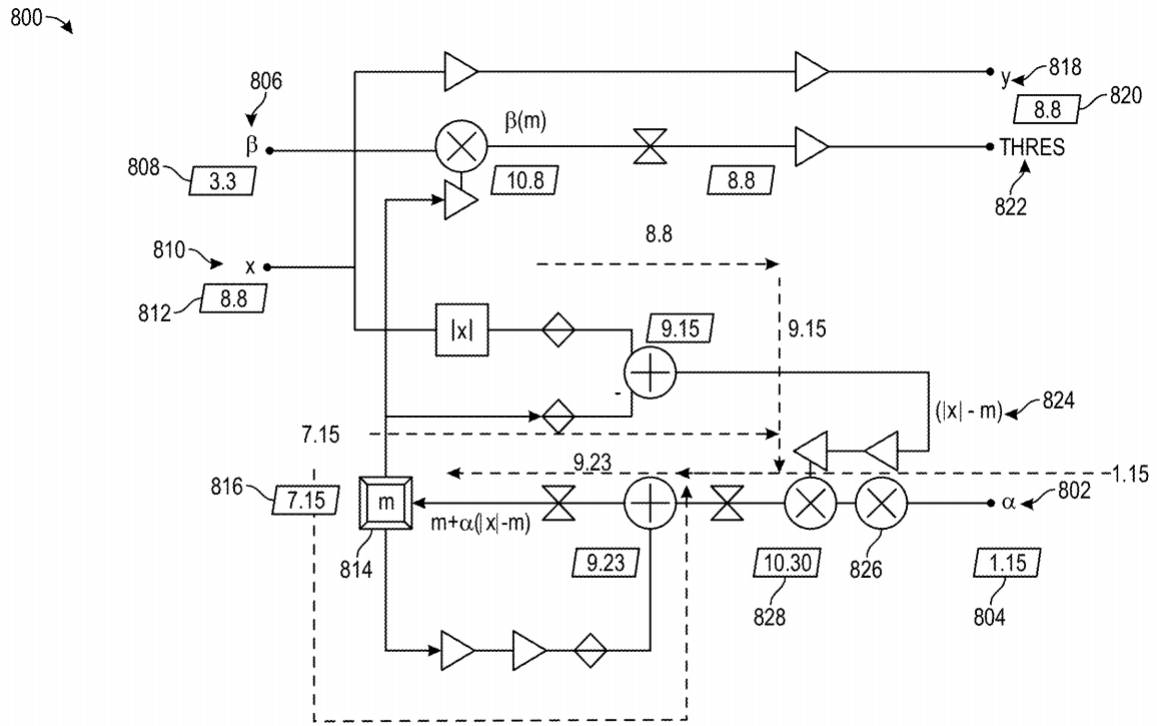
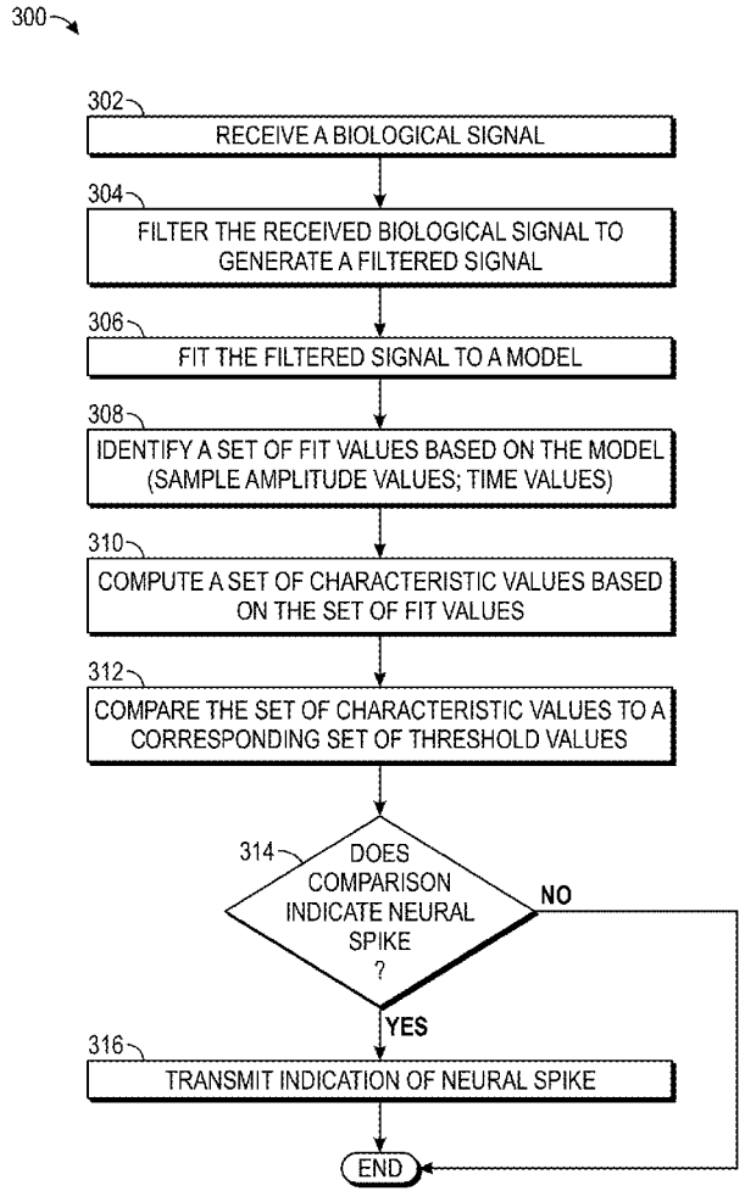


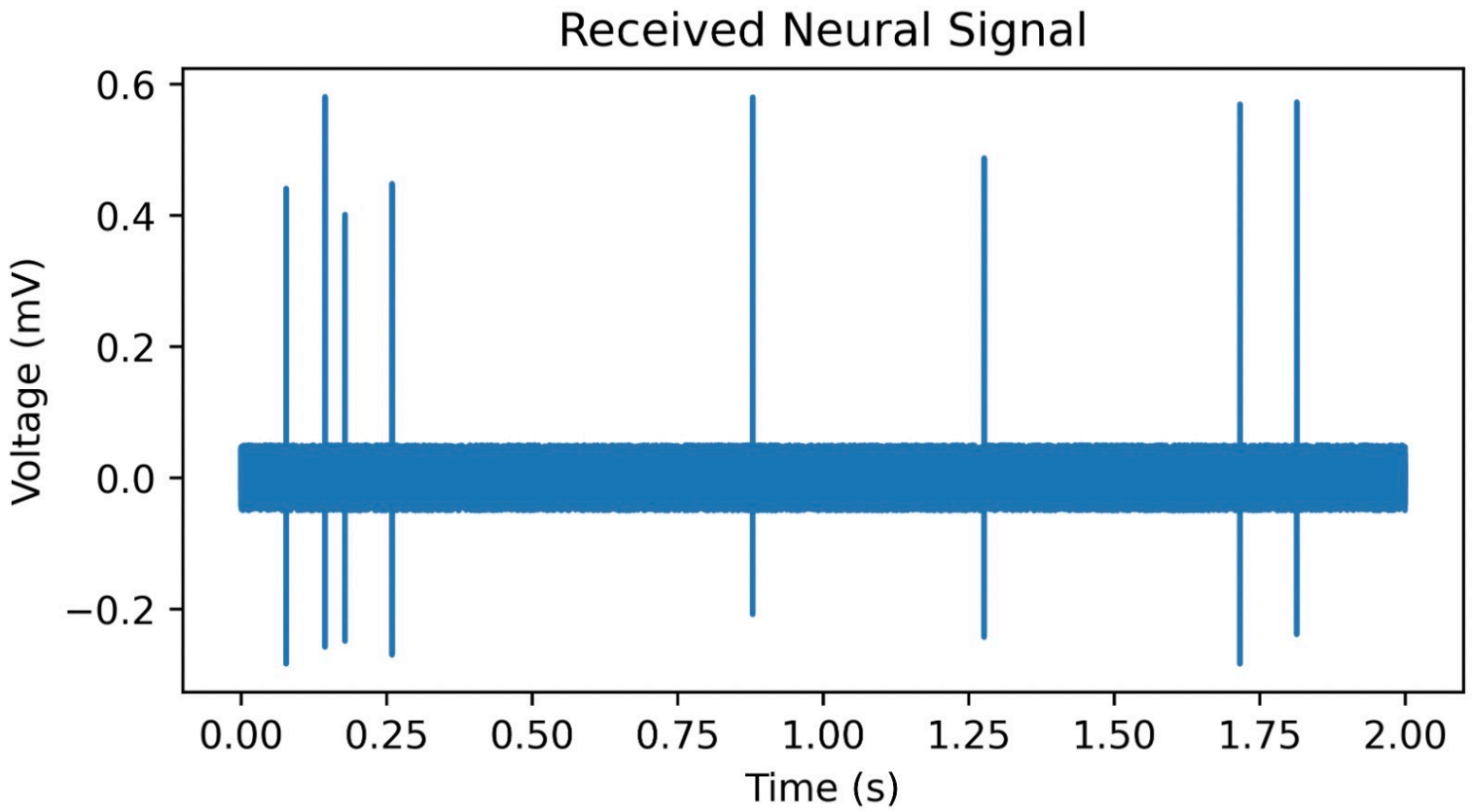
FIG. 8

Overall Procedure



1. Receive Biological Signal

Receive neurological voltage signal from electrode leads to IC chip



2. Filter Signal

Filtering - **Get rid** of **noisy** signals, **retain** a **subset** of signals

This patent uses a **16-bit digital Infinite Impulse Response (IIR) Butterworth 2nd-order Bandpass Filter** (Fig. 9)

Equations: $y = xb_0 + w_1$

$$\widehat{w_1} = w_2 + xb_1 - ya_1$$

$$\widehat{w_2} = xb_2 - ya_2$$

x - original sample, y - filtered signal
a1, a2, b0, b1, b2 - user-configurable params
w1_hat, w2_hat - intermediate states

This algorithm **ALWAYS runs** while the **device is on!**

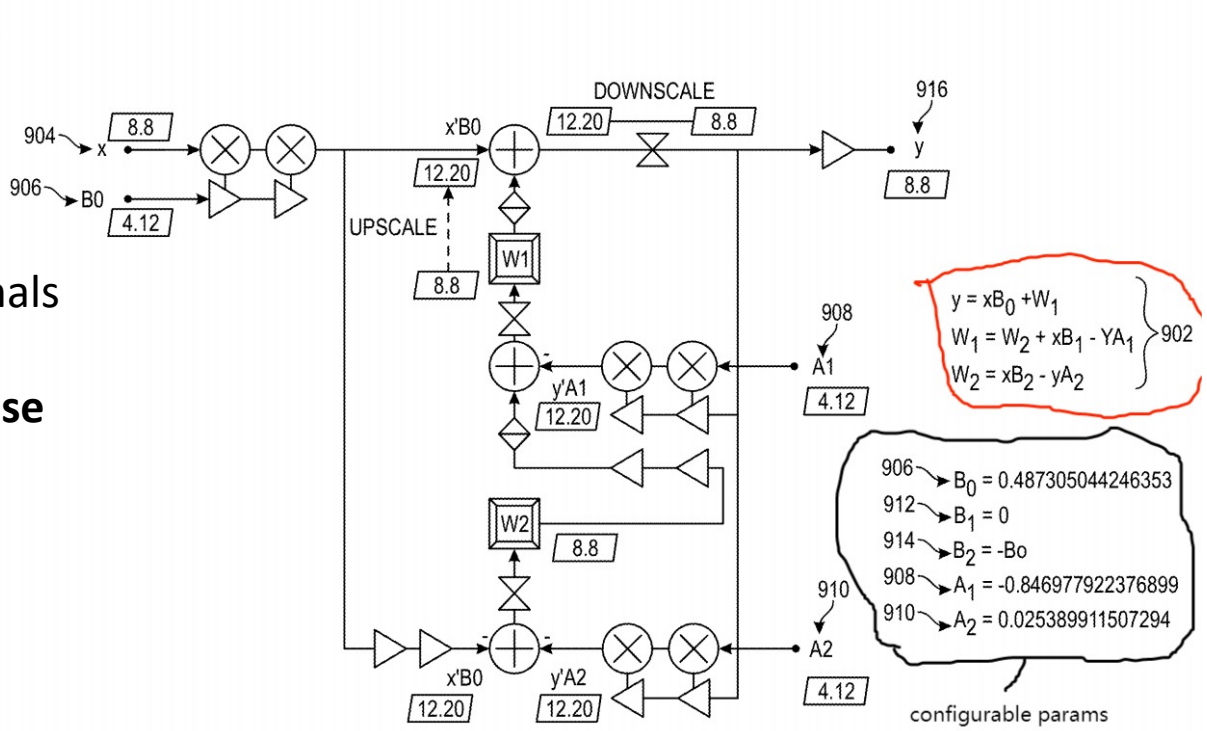
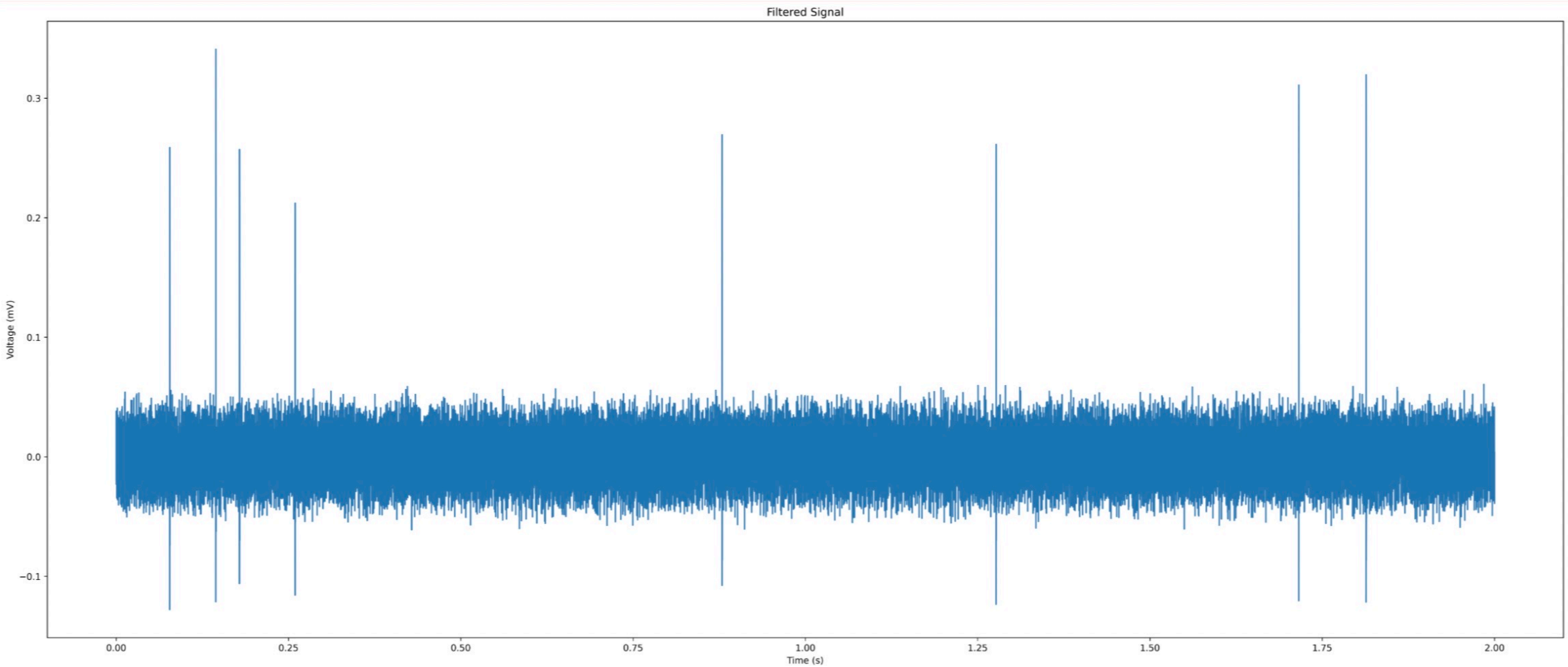


FIG. 9

multiplication (x), addition (+), conversion (hourglass)
division - assymetric multipliers (used with cycles)
decided by smallest operand (16b)


Filtered Signal
Voltage (mV)
0.3
0.2
0.1
0.0
−0.1
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Time (s)

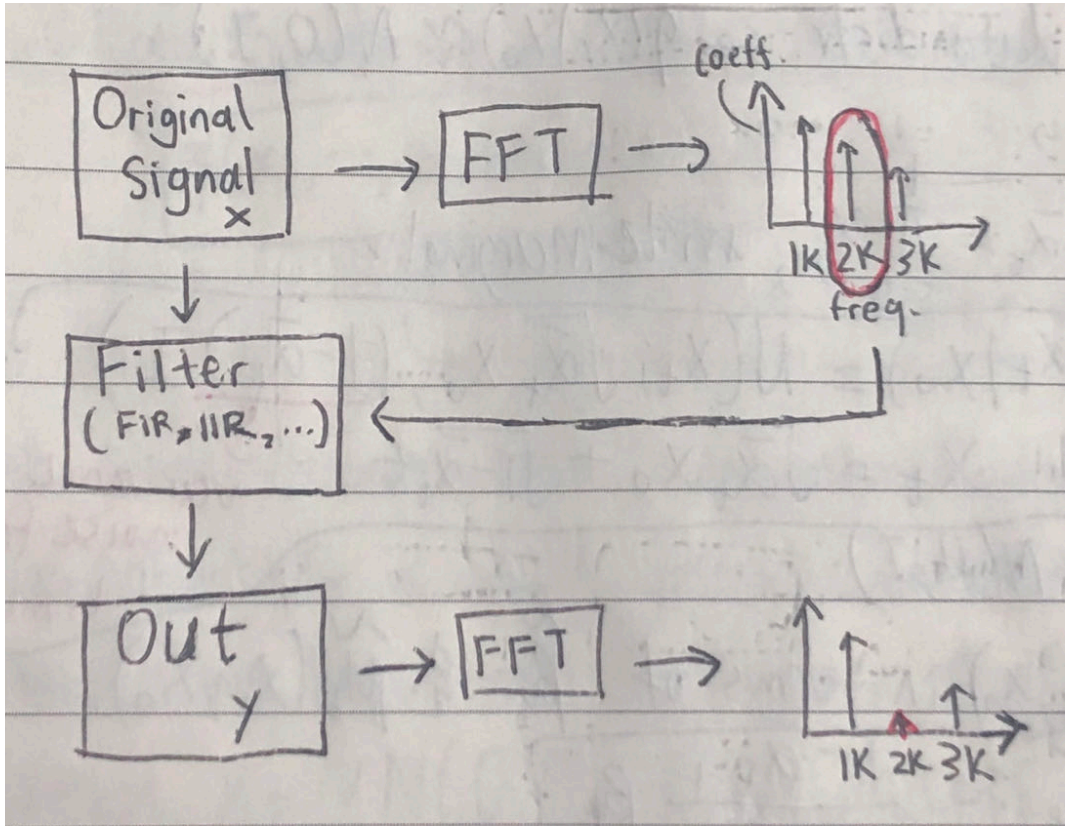
For Deeper Intuition of Filtering Signals

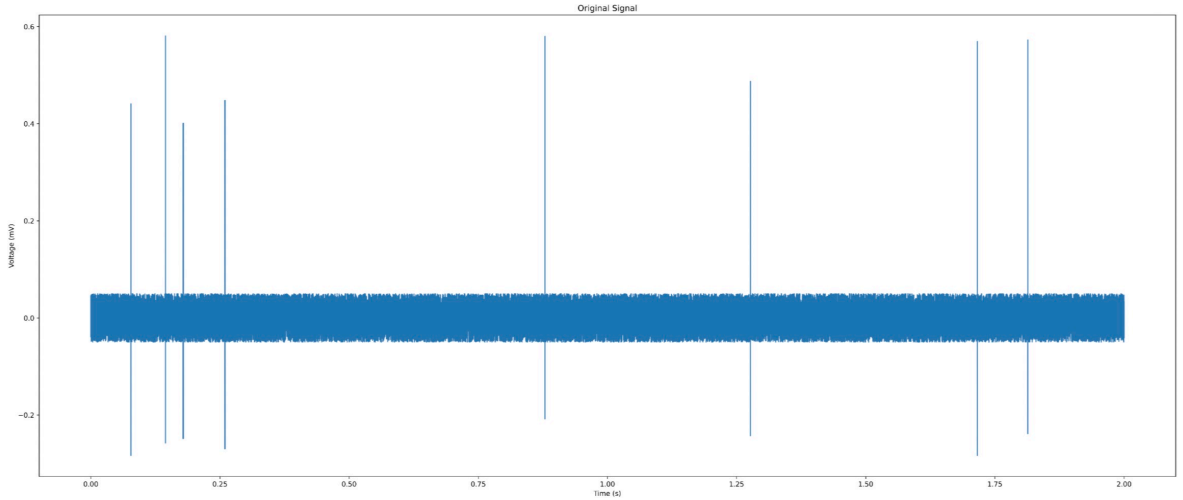
1. **Original** Signal
2. **Fast-Fourier Transform (FFT)** of original signal

Determine **frequency band [low_cut, high_cut]** based on **FFT** (which **principal components** do we **keep/remove**?)

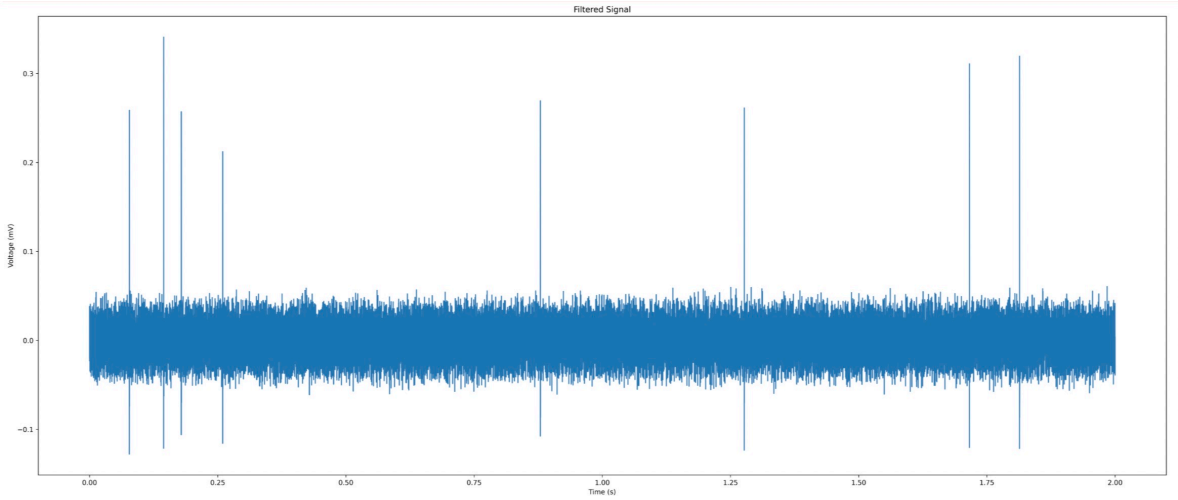
3. **Filtered** signal
4. **Fast-Fourier Transform (FFT)** of **filtered** signal

Check if **filtered signal** & **FFT plots** are what is desired

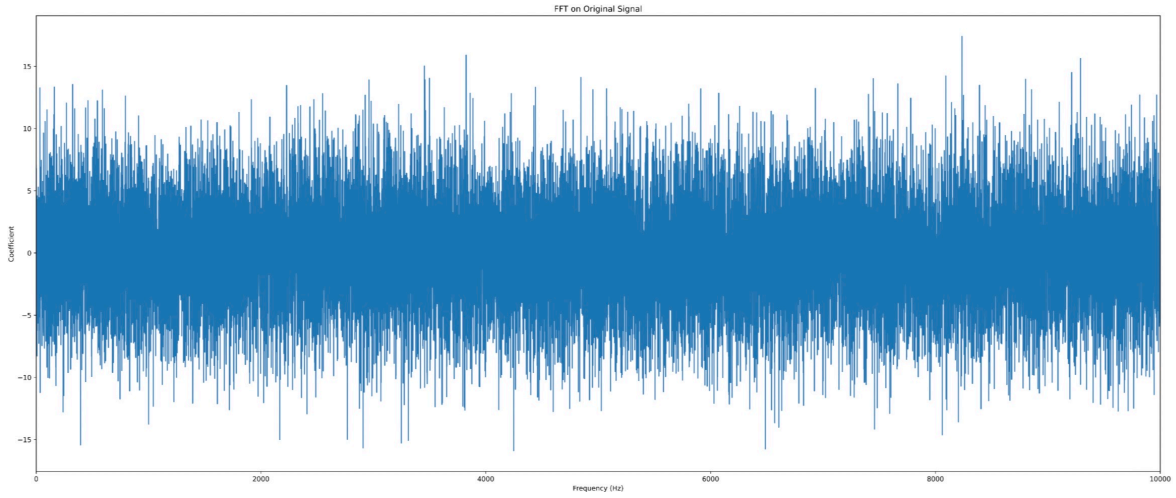




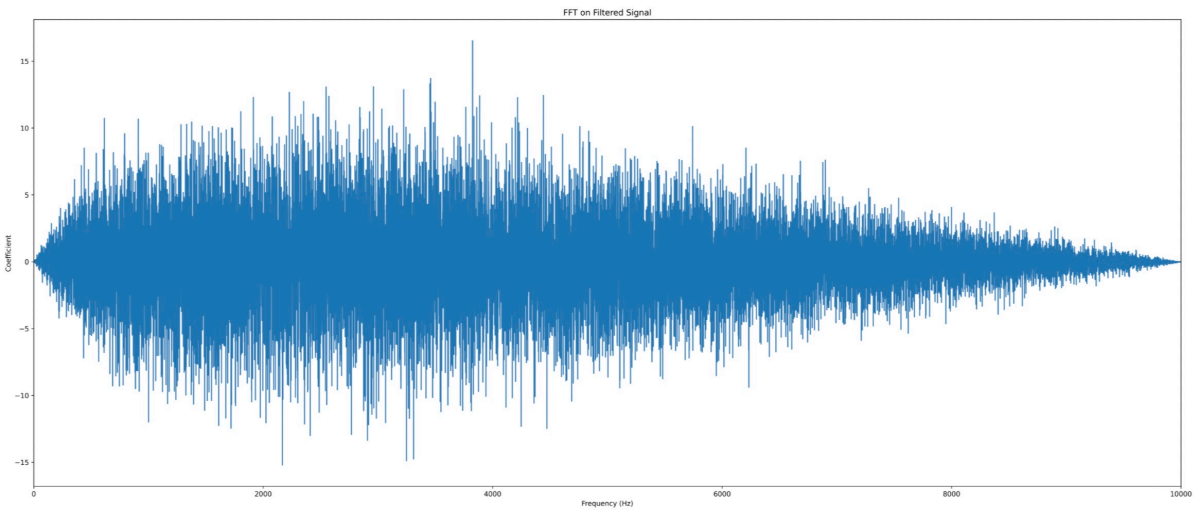
1. Original



3. Filtered



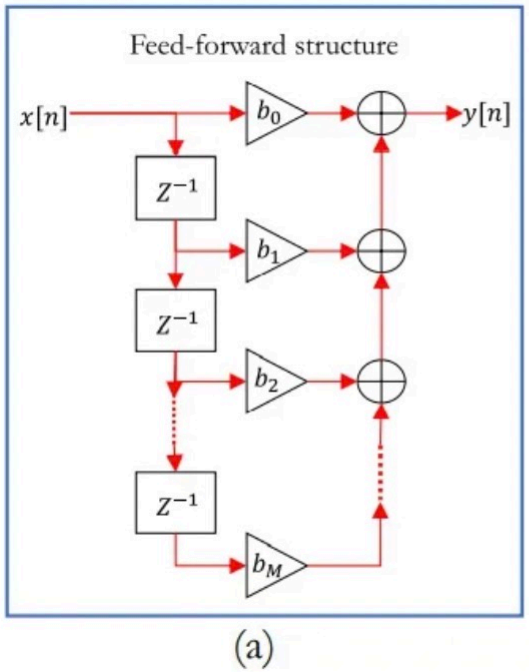
2. FFT on original



4. FFT on filtered

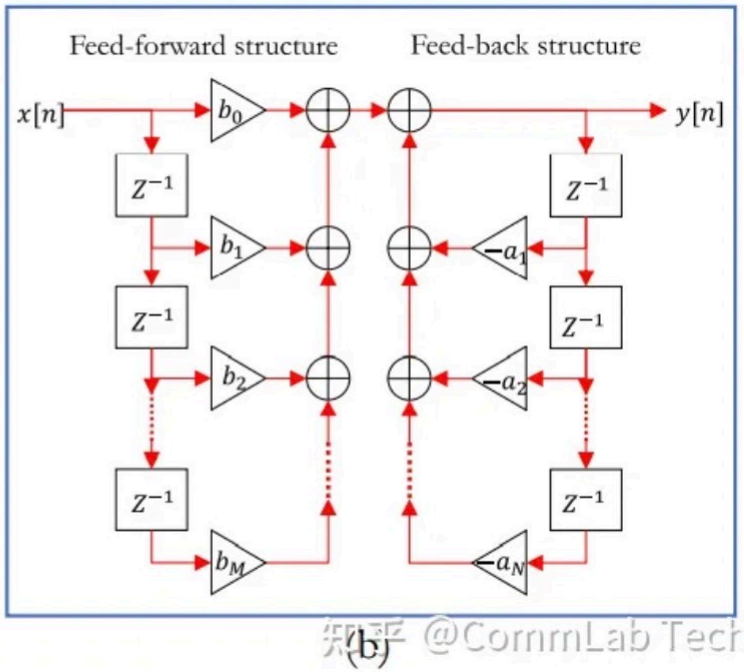
Filtering Basics:
Finite Impulse Response (FIR) vs Infinite Impulse Response (IIR)

Finite Impulse Response (FIR)



Feed-forward only (no loops)
Used in **engineering**
Easy to implement in **hardware** due to **high precision**

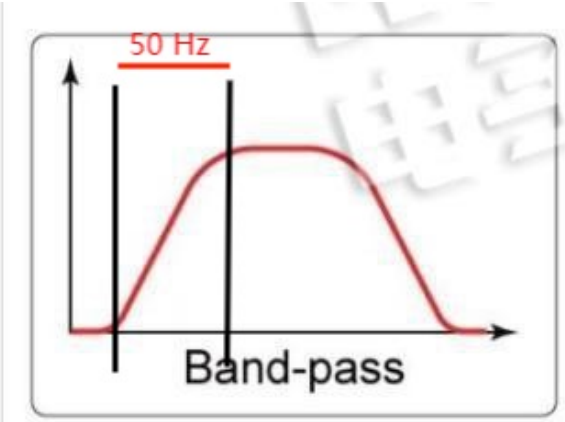
Infinite Impulse Response (IIR)



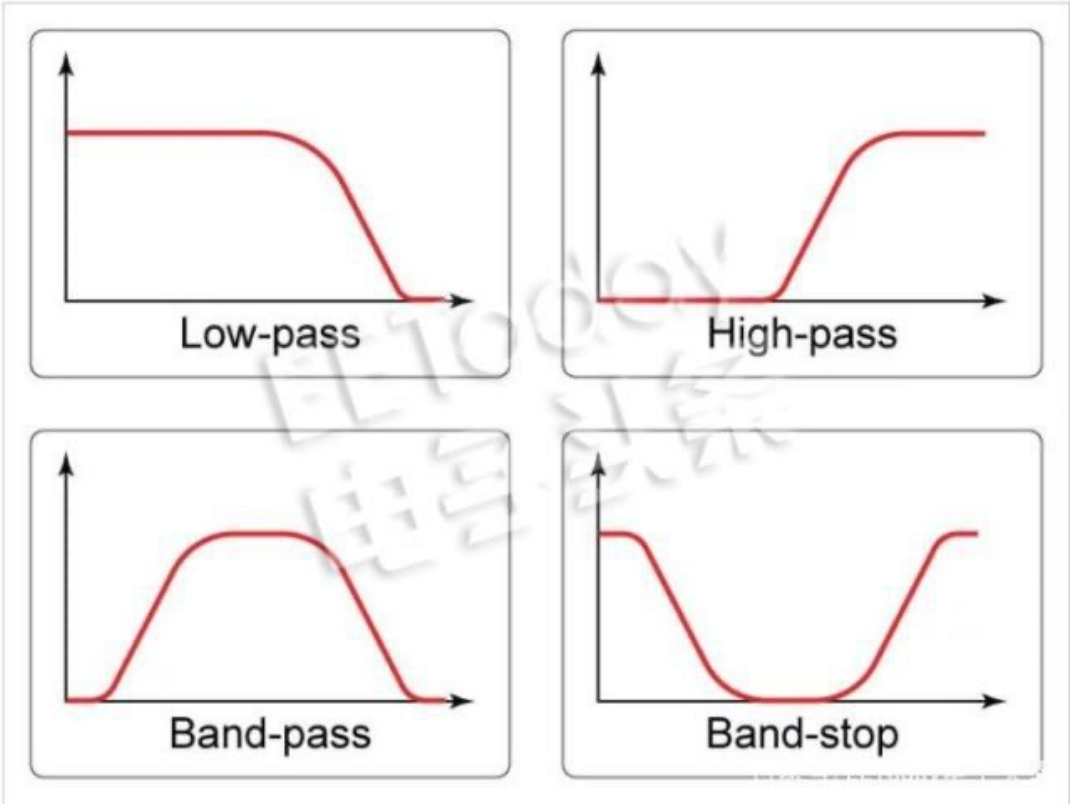
Includes **feed-back** structure (**loops**)
Used in **theoretical research**
Very **difficult to implement** in **hardware**
(usually used in **software** such as **Python**)

Filtering Basics:
Low-Pass, High-Pass, Band-Pass, Band-Stop

- **Low-pass**: Keep **low frequency** signal portions
- **High-pass**: Keep **high frequency** signal portions
- **Band-pass**: Keep signal portions in a **configured frequency band**
- **Band-stop**: **Remove** signal portions that are in **configured frequency band**



Note: Make sure that **configured bands** are **wide enough**!

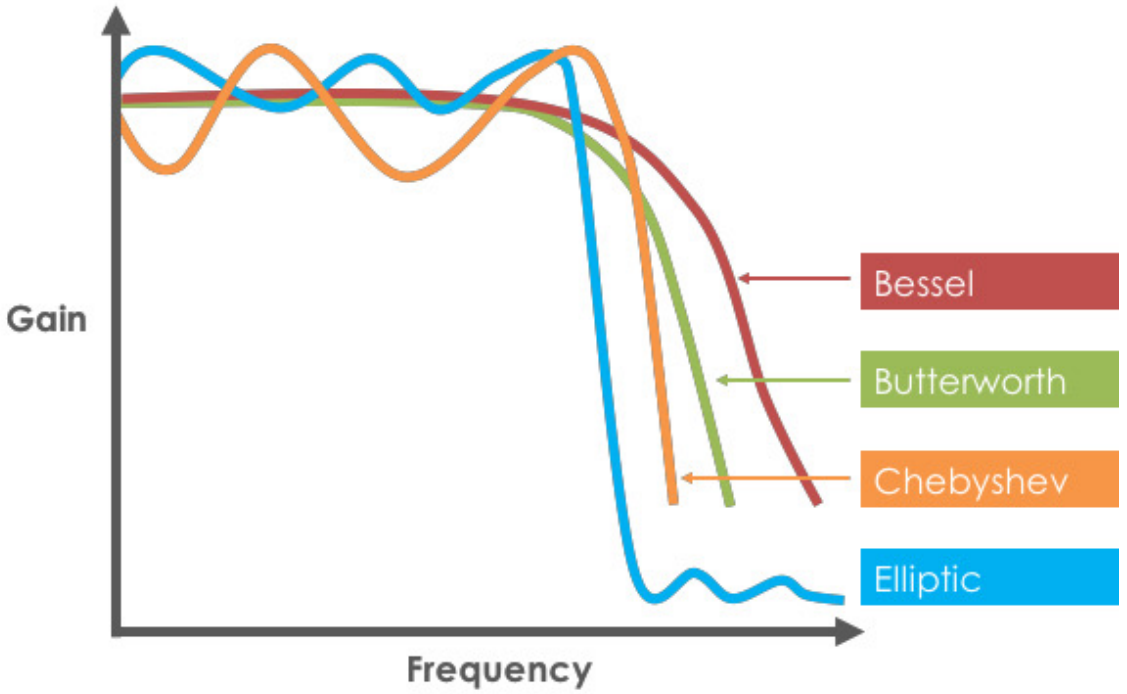


Filtering Basics:
Bessel, Butterworth, Chebyshev, Elliptic

- **Butterworth** – **maximally flat frequency** response in **passband**
- **Chebyshev** – **steeper roll-off** with **ripple** in passband/stopband
- **Bessel** – **maximally flat group delay**
- **Elliptic** – **steeper roll-up with ripple** in **passband** and **stopband**, but **steepest roll-off** achievable in **given order**

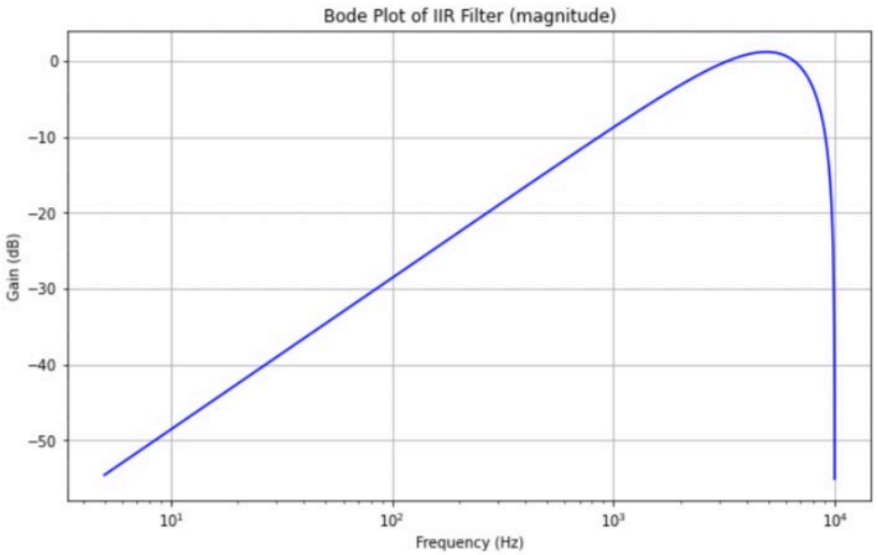
Patent uses **Butterworth** filter:

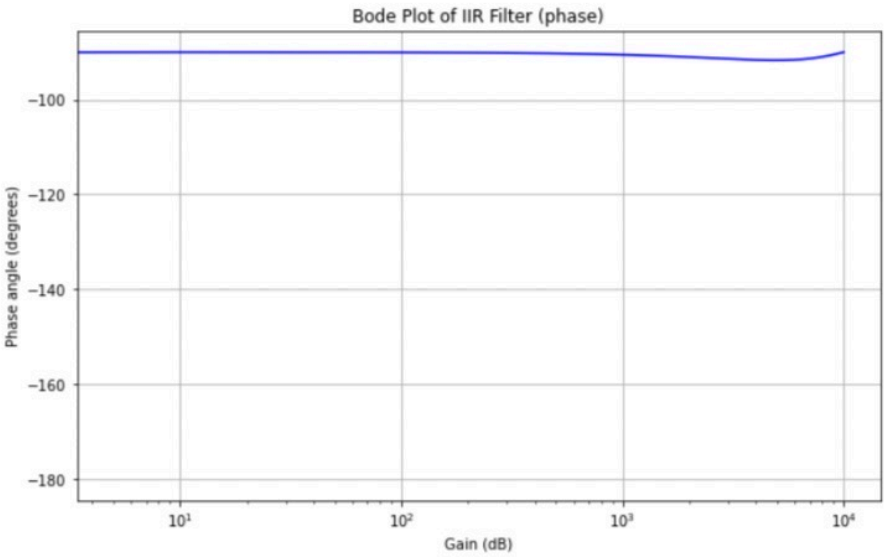
- **Smooth frequency response** in **passband** (for **integrity** of neural signals)
- Roll-off not critical
- **Minimal phase shift**
- **Computationally efficient**
- Commonly used in **audio** applications, **neural signal processing**



Filtering Basics: **Bode Plot**

- Used for **amplitude and frequency analysis**
- Obtained from **transfer function params b (b0, b1, b2)** and **a (a1, a2)**
- **2 separate plots**: **magnitude** and **phase**
- Used in **engineering**
- **Nyquist** plot- used in **research/academia** (magnitude + phase plots all in one)





The **transfer function formula** is given as:

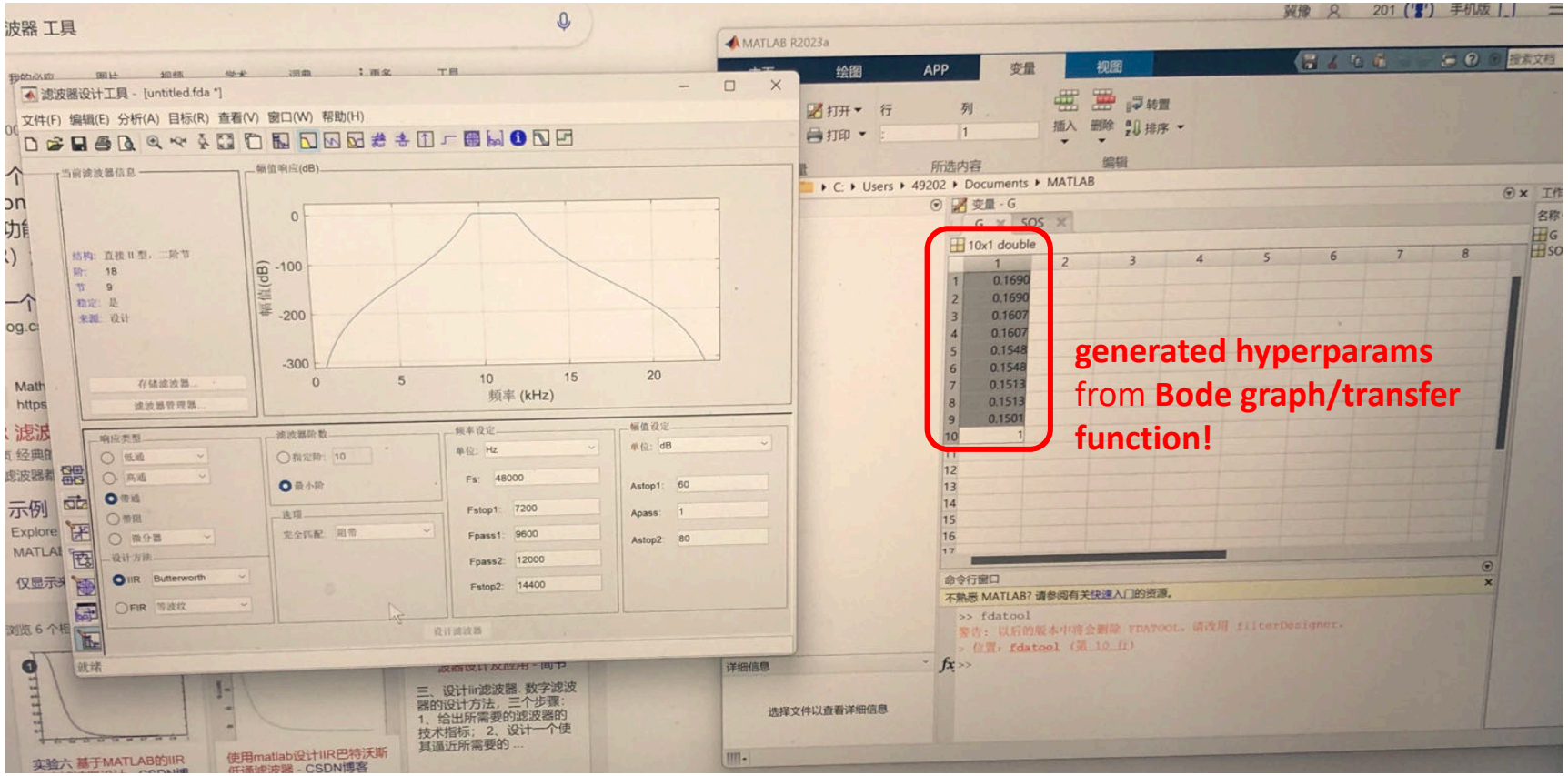
$$H(s) = \sum_{i=0}^{N} b[N-i]s^i \,/\, \sum_{j=0}^{M} a[M-j]s^j$$

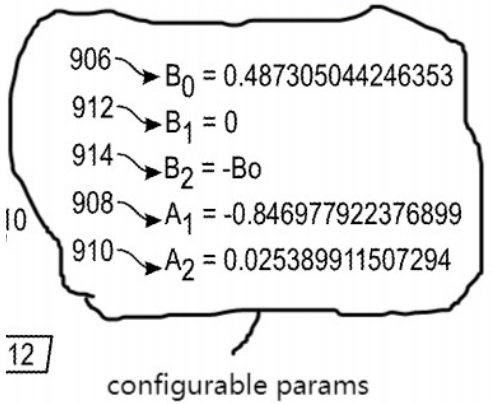
where $b = b_0, b_1, \ldots, b_N$ and $a = a_0, a_1, \ldots, a_M$

```
b: [ 0.48730504  0.         -0.48730504]
a: [ 0.         -0.84697792  0.02538991]
```

$$\frac{0.4873s^2 - 0.4873}{-0.847s + 0.02539}$$

Filtering: Where do the params come from?





Normally, you would want to **obtain the params b, a** from the **IIR filter itself** to plot the Bode graph, instead of copying parameters.

3. Smooth Signal

- **Smoothing** - **remove system noise** for **unbiased estimation** and **better quality data**
- Uses **modern control theory**
- **Exponential Smoothing** - captures **non-linearity (e.g., spikes)** in data

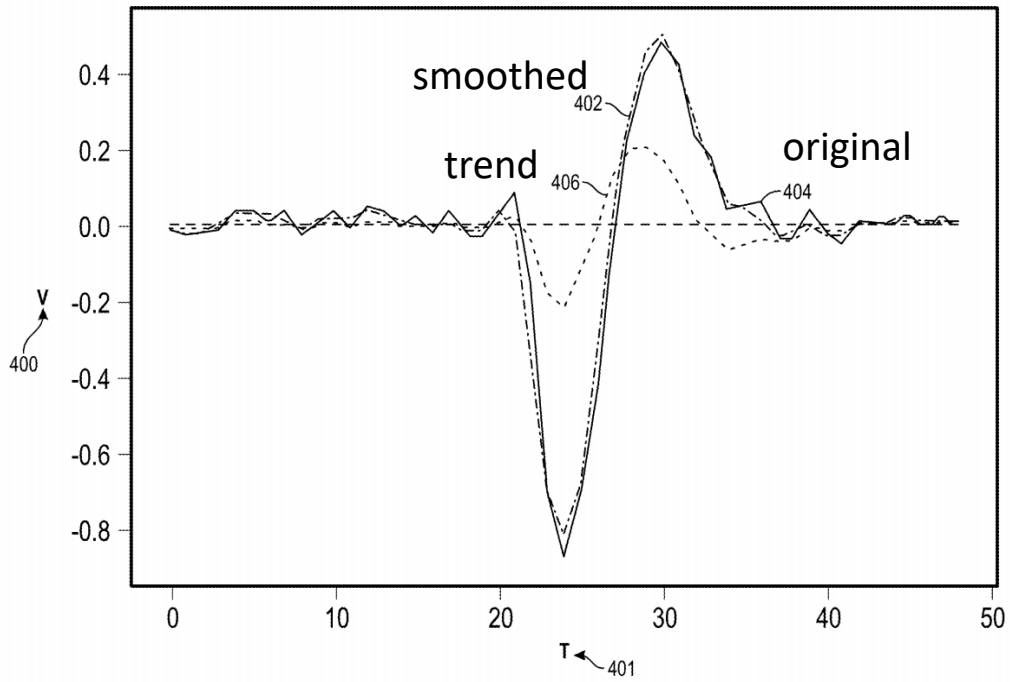
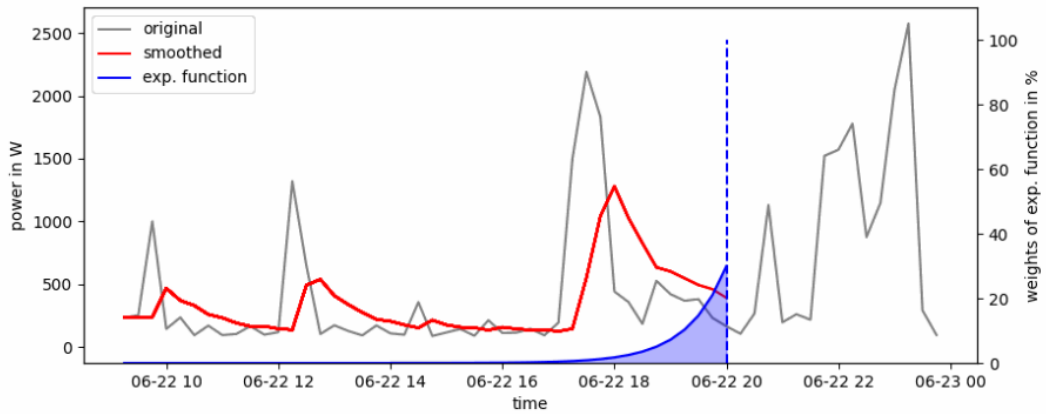
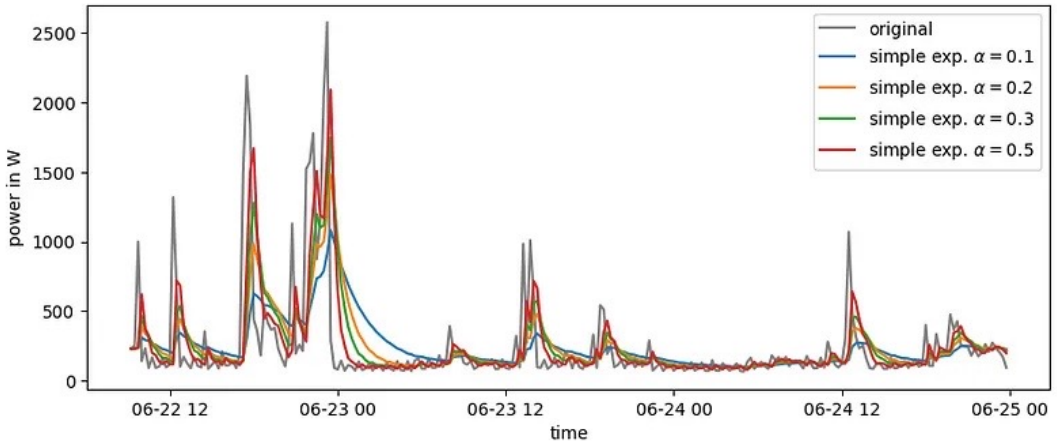


FIG. 4A



Single Exponential Smoothing

- **More recent** data points - **bigger weight** than the ones further in the past
- **No seasonality or trends**
- 1 hyperparam (α from 0-1) to control **how much emphasis** to put on **recent data points**
- Used in **short-term prediction** (older history gets eliminated)
- Prone to **lagging**



$$\hat{y}_t = \begin{cases} \hat{y}_t = y_t & \text{if } t = 0 \\ \alpha y_t + (1-\alpha)\hat{y}_{t-1} & \text{otherwise} \end{cases}$$

where α is the smoothing factor

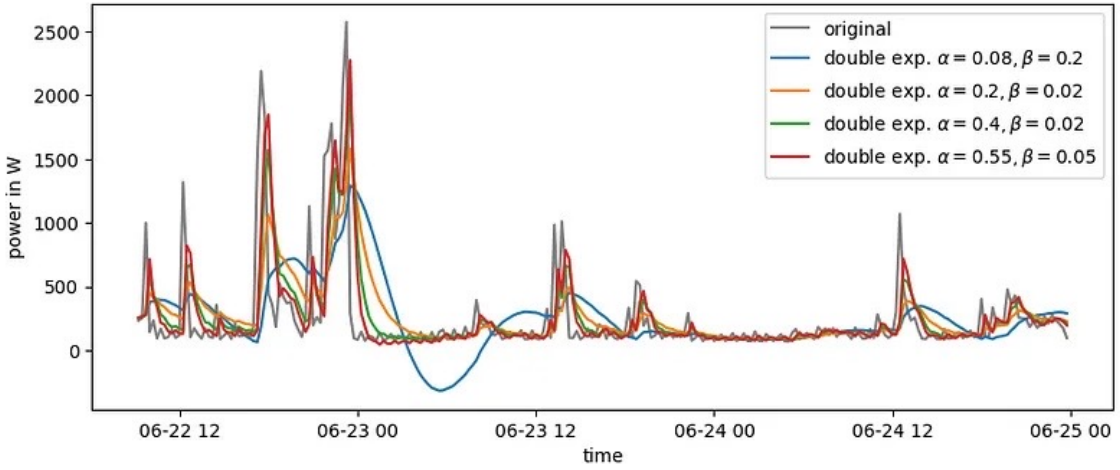
(**bigger α -- smaller emphasis** on **past data points**, **smaller smoothing**)

ts eliminated)

one to lagging

Double (Linear) Exponential Smoothing

- Incorporates **level** (smoothed value of signal) **and trend** (how data **progresses over time**)
- Normally **2 hyperparams** (in range 0-1):
  - α - controls **level** of data
  - β - controls **trend** of data
- **Trend** can be **additive (linear)** or **multiplicative (exponential)**
- Most common: **Holt Double Exponential Smoothing** (**additive trend shown on right**)



$$\hat{y}_t = l_t + r_t$$

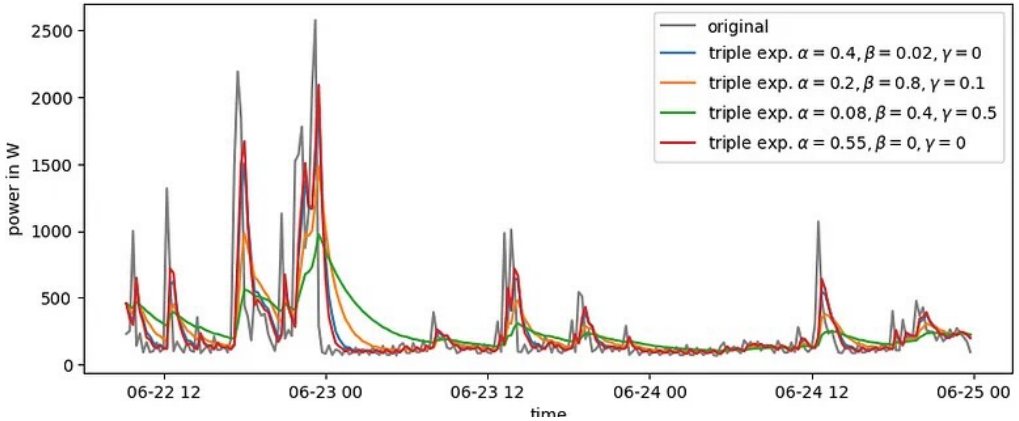
$$l_t = \begin{cases} y_t & \text{if } t = 0 \\ \alpha y_t + (1-\alpha)(l_{t-1} + r_{t-1}) & \text{otherwise} \end{cases}$$

$$r_t = \begin{cases} y_{t-1} - y_t & \text{if } t = 0 \\ \beta(l_t - l_{t-1}) + (1-\beta) r_{t-1}) & \text{otherwise} \end{cases}$$

where l is the level and r is the trend.

Triple Exponential Smoothing

- Incorprates **level, trend, AND seasonality** (**regular, predictable** changes occurring **every time period**)
- Normally **3 hyperparams** (in range 0-1):
  - α - controls **level** of data
  - β - controls **trend** of data
  - γ - controls **seasonality** of data
- **Seasonality** can be **additive or multiplicative**
- Most common: **Holt-Winters Exponential Smoothing (additive seasonality + trend shown in right)**



$$\hat{y}_{t+k} = l_t + kr_t + s_{t+k-M}$$

$$l_t = \alpha(y_t - s_{t-M}) + (1-\alpha)(l_{t-1} + r_{t-1})$$

$$r_t = \begin{cases} \frac{\sum_{i=1}^{M} y_{M+i} - y_i}{M} & \text{if } t = 0 \\ \beta(l_t - l_{t-1}) + (1-\beta)r_{t-1}) & \text{otherwise} \end{cases}$$

$$s_t = \gamma(y_t - l_t) + (1-\gamma)s_{t-M}$$

where l_t is the level estimate for time t, k is the number of forecasts into the future,
r_t is the trend estimate at time t, M is the number of seasons.
s_t is the seasonal estimate at time t,

Brown's Double Exponential Smoothing (used in patent)

- Uses only **one (1) hyperparam** (α) to tune **BOTH level and trend**
- **No seasonality** involved (spikes may occur at **ANY** time)
- **Used in this patent**
- Notebook uses $\alpha = 0.1$

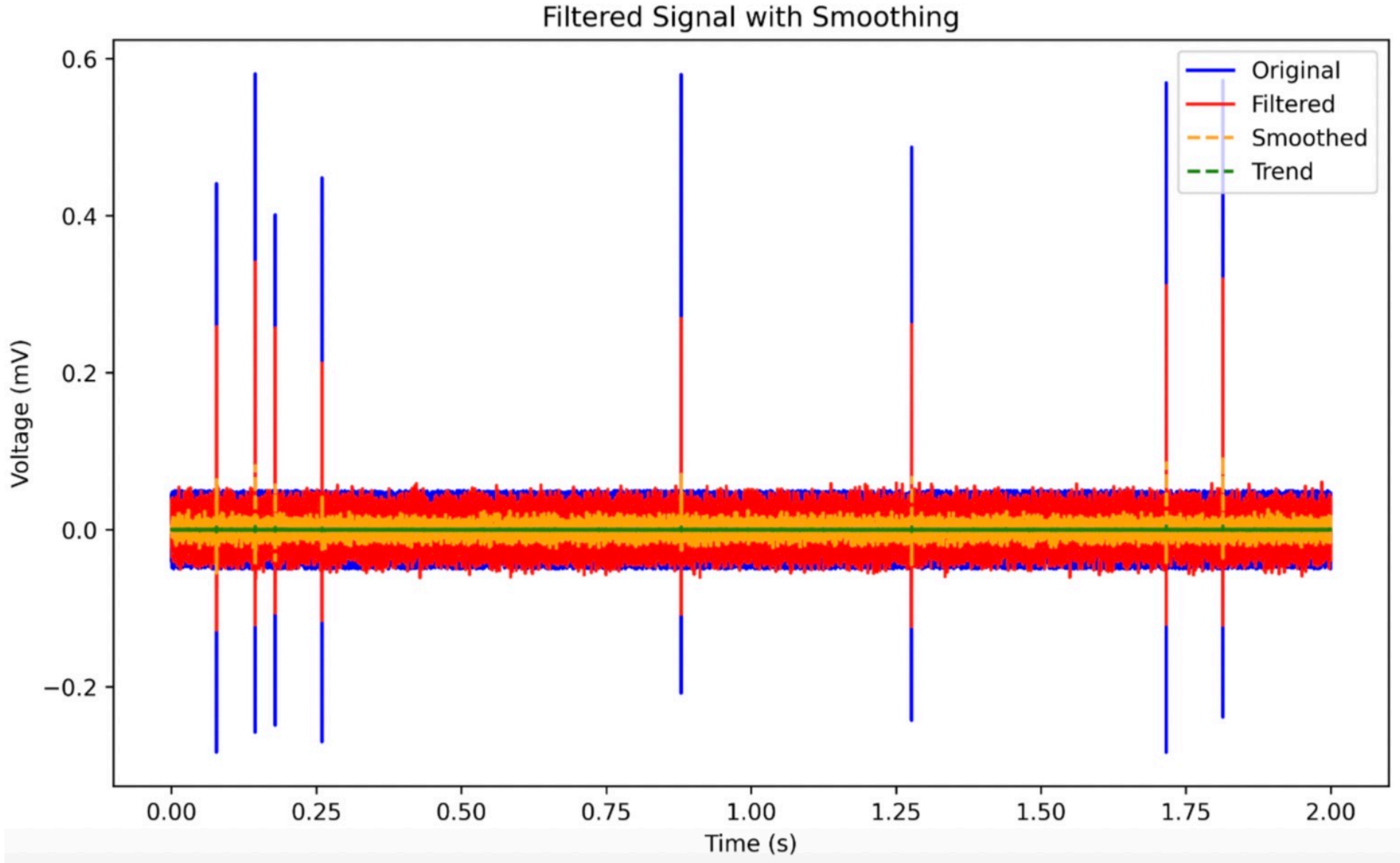
Single exponential smoothing: $S'(t) = \alpha Y(t) + (1 - \alpha)S'(t-1)$

Double exponential smoothing: $S''(t) = \alpha S'(t) + (1 - \alpha)S''(t-1)$

Forecast: $\hat{y}_{t+1} = l_t + r_t$

where $l_t = 2S'(t) - S''(t)$ (estimated level at time t)

$r_t = \frac{\alpha}{1-\alpha}(S'(t) - S''(t))$ (estimated trend at time t)


Filtered Signal with Smoothing
Original
Filtered
Smoothed
Trend
Voltage (mV)
0.6
0.4
0.2
0.0
−0.2
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Time (s)

4. Identify fit values

- i.e., identify **local maxima and minima** of **filtered signal** in **each time window**

- Can (optionally, but not needed) plot a polynomial function to identify fit values

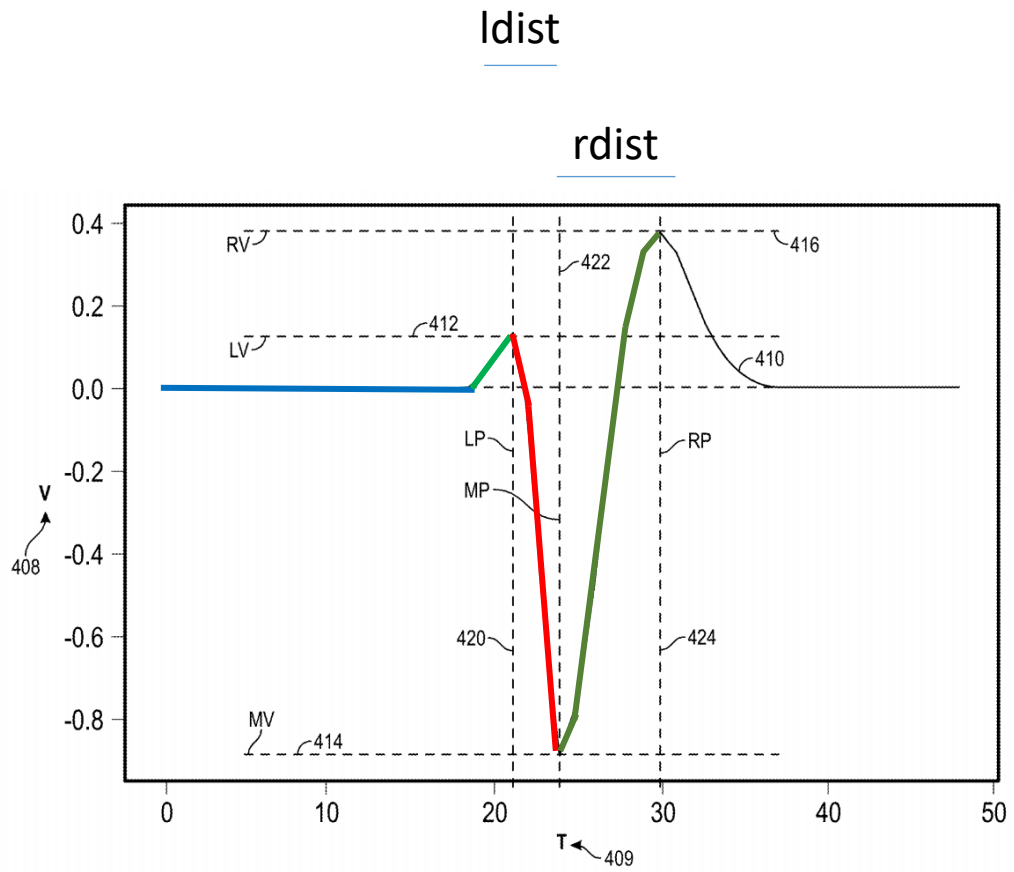


FIG. 4B

Identify local minima/maxima per time window

- For each fixed time window (blackout period), want to find the **local min and max**
- Notebook uses **40ms blackout period**
- Absolute deviation of the signal accumulated to smoothed function for fixed time period

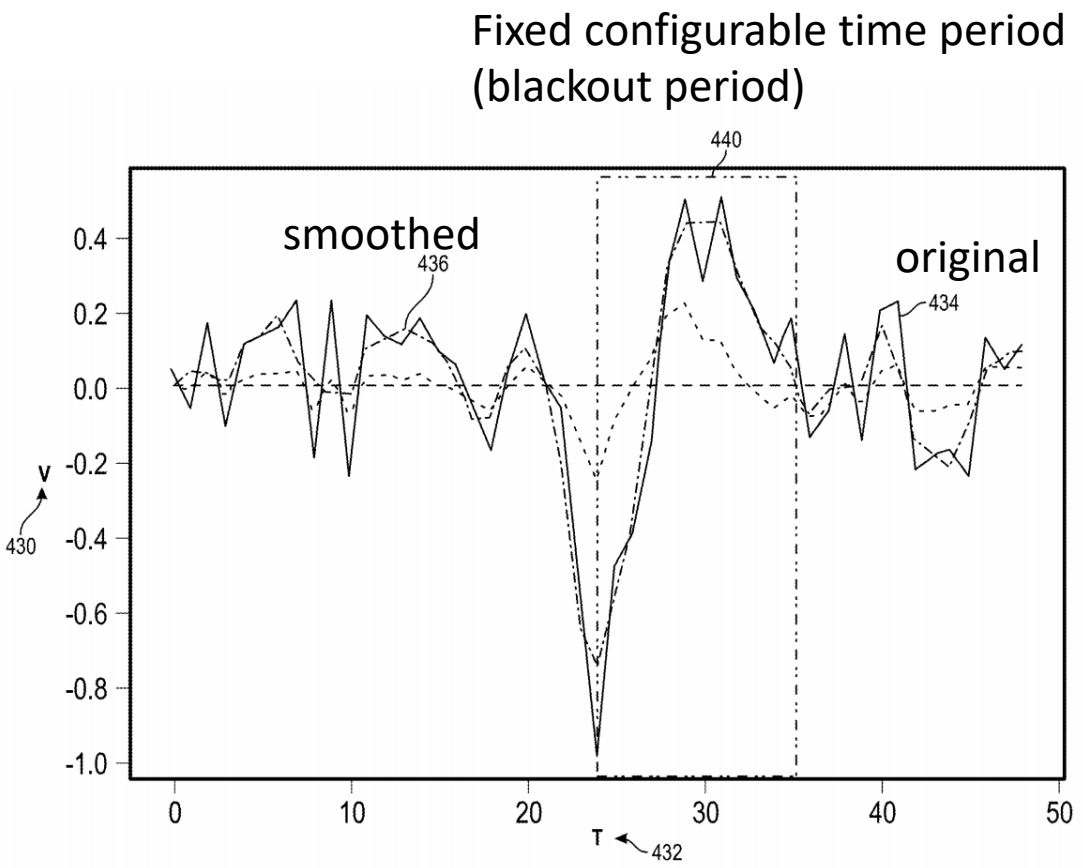
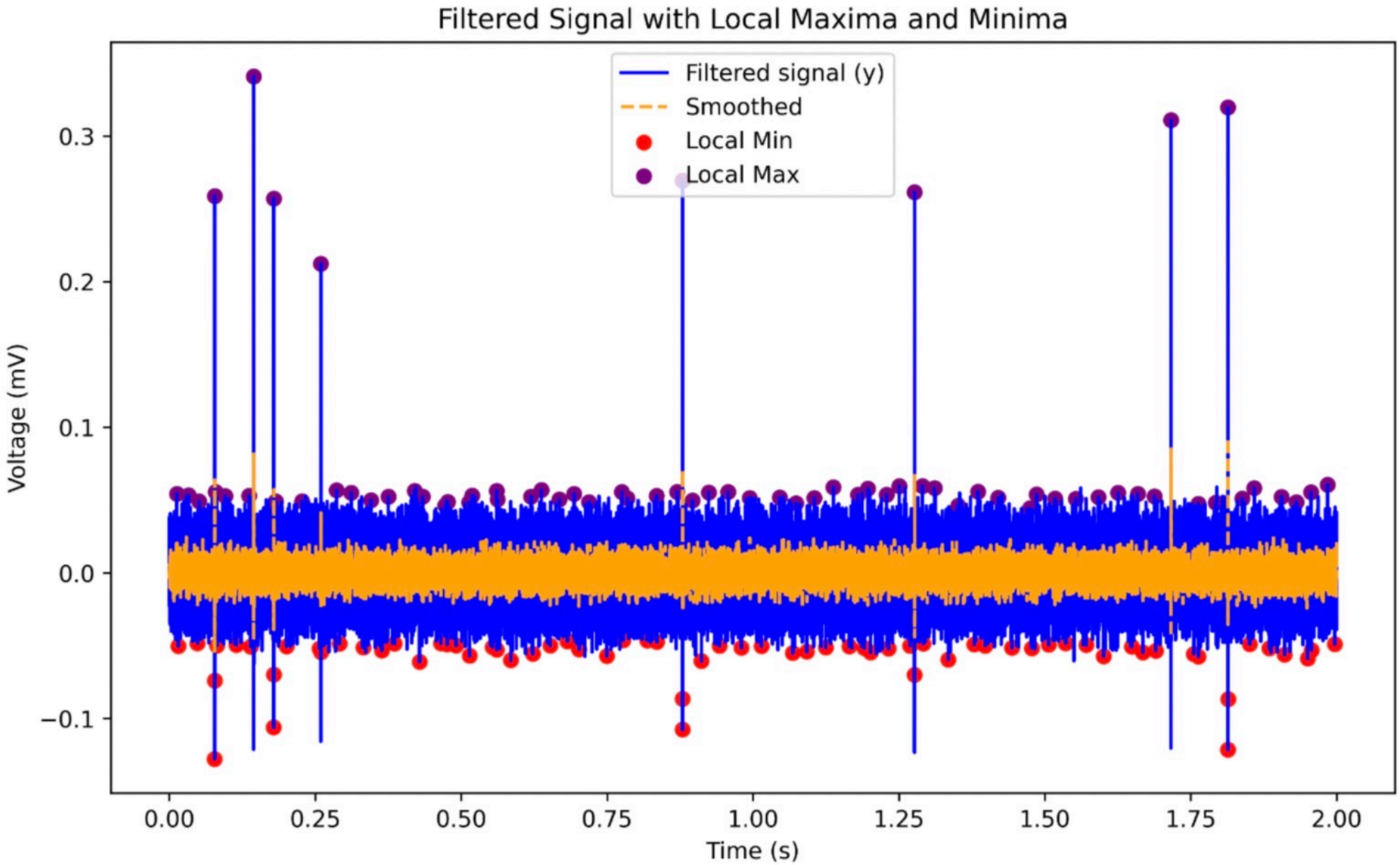


FIG. 4C


Filtered Signal with Local Maxima and Minima
Filtered signal (y)
Smoothed
Local Min
Local Max
Voltage (mV)
0.3
0.2
0.1
0.0
−0.1
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Time (s)

5. Compute Characteristic Values

The **characteristic values** (in a **configured time window**) are shown as:

$ldist = mp - lp$ (time elapsed between first local maximum and local minimum)

$rdist = rp - mp$ (time elapsed between local minimum and second local maximum)

$ratio = \frac{rv}{|mv|}$ (relative voltage ratio of second local maximum and local minimum voltage magnitude)

$asym = \frac{lv}{rv}$ (meausre of asymmetric property - $asym = 1$ means the left & right local maxima are symmetric)

$cost = \frac{esterr}{|mv|}$ (ratio between second local maximum voltage and local minimum voltage magnitude)

where lp, mp, rp are the **left, middle, and right timestamps** and lv, mv, rv are the **left, middle, and right voltage values (shown in Fig. 4B above)**.

$esterr$ is given as the **mean absolute deviation (MAD) estimate**.

We sample $[lp, rp]$ as our **time window**.

In our **notebook**, we sample **local max-local min-local max (lp, mp, rp)** as our **(non-fixed length) time window** for **characteristic value calculation**. The **local min and local max** should be computed in the **last step** for each **fixed-length time window**.

Mean Absolute Deviation (MAD)

- Original MAD estimation formula:

$$\hat{m} = m + \alpha(|x| - m),$$

where m is the **mean** over **smoothed** signal, α is the **user-configured update multiplier** (set to 0.0002)

- Since we are taking the mean over a **time interval**, our formula changes as follows:

$$\hat{m} = \frac{\sum_{t=t_1}^{t_2} m+\alpha(|x_t|-m)}{\sum_{t=t_1}^{t_2} |x_t|-m},$$

where m is the mean of the smoothed series **over a time interval**

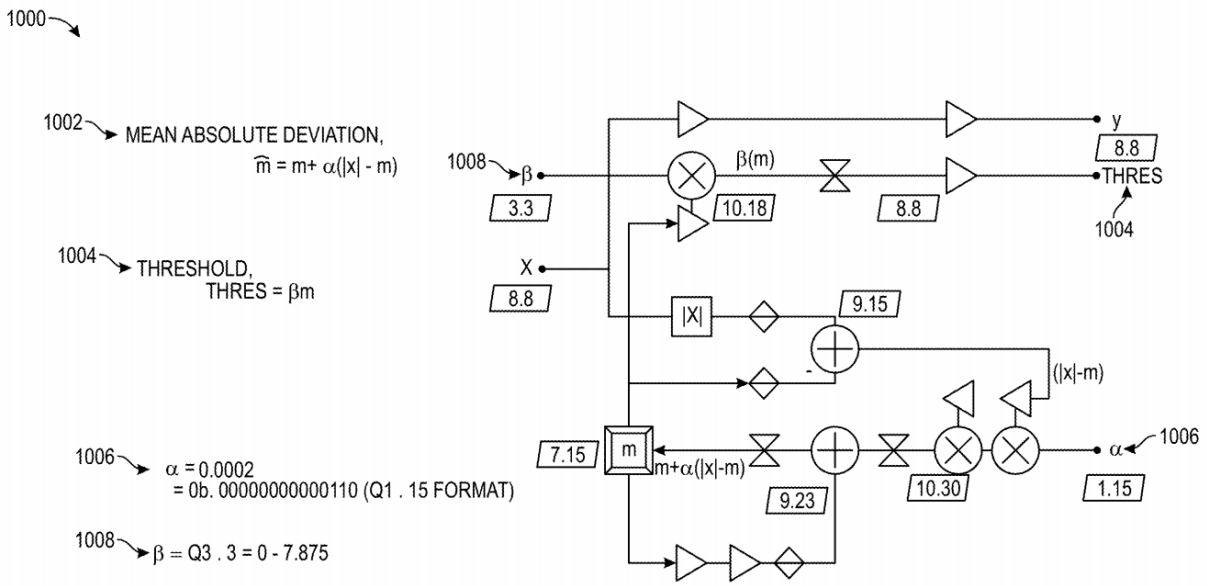


FIG. 10

Threshold Calculation

- The threshold $thres$ is given as:

$$thres = \beta * m,$$

where $\boldsymbol{\beta}$ is a **scale factor constant** in range **0 - 7.875** (given in **Q3.3 format** - 3 bits integer, 3 bits fraction, as a **multiple of 0.125**)

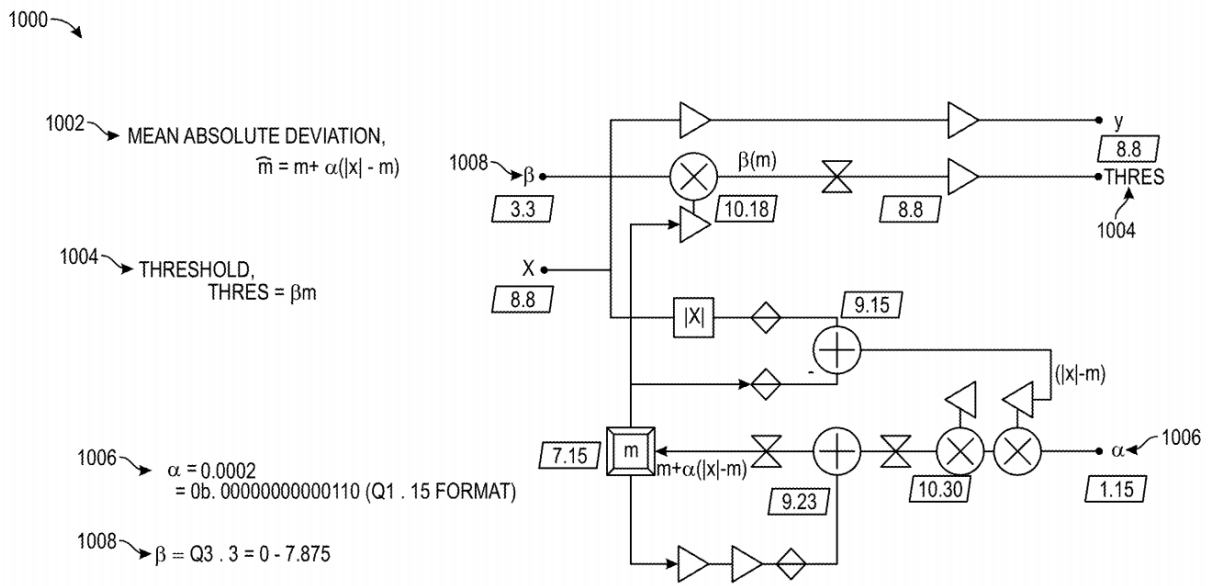


FIG. 10

6. Determine Threshold Values

- Can plot characteristic values separately to determine optimal threshold values
- Plots can be shown in notebook
- Final thresholds determined in figure on the right

```
# Set threshold values here
MIN_RATIO = 1.33
MAX_ASYM = 1
MAX_COST = 2
MIN_THRES = -0.5
MAX_THRES = 0.1
MAX_LDIST = 0.2
MIN_RDIST = 5e-5
MAX_RDIST = 0.2
```

7. Determine whether spike is detected + timestamps of spikes

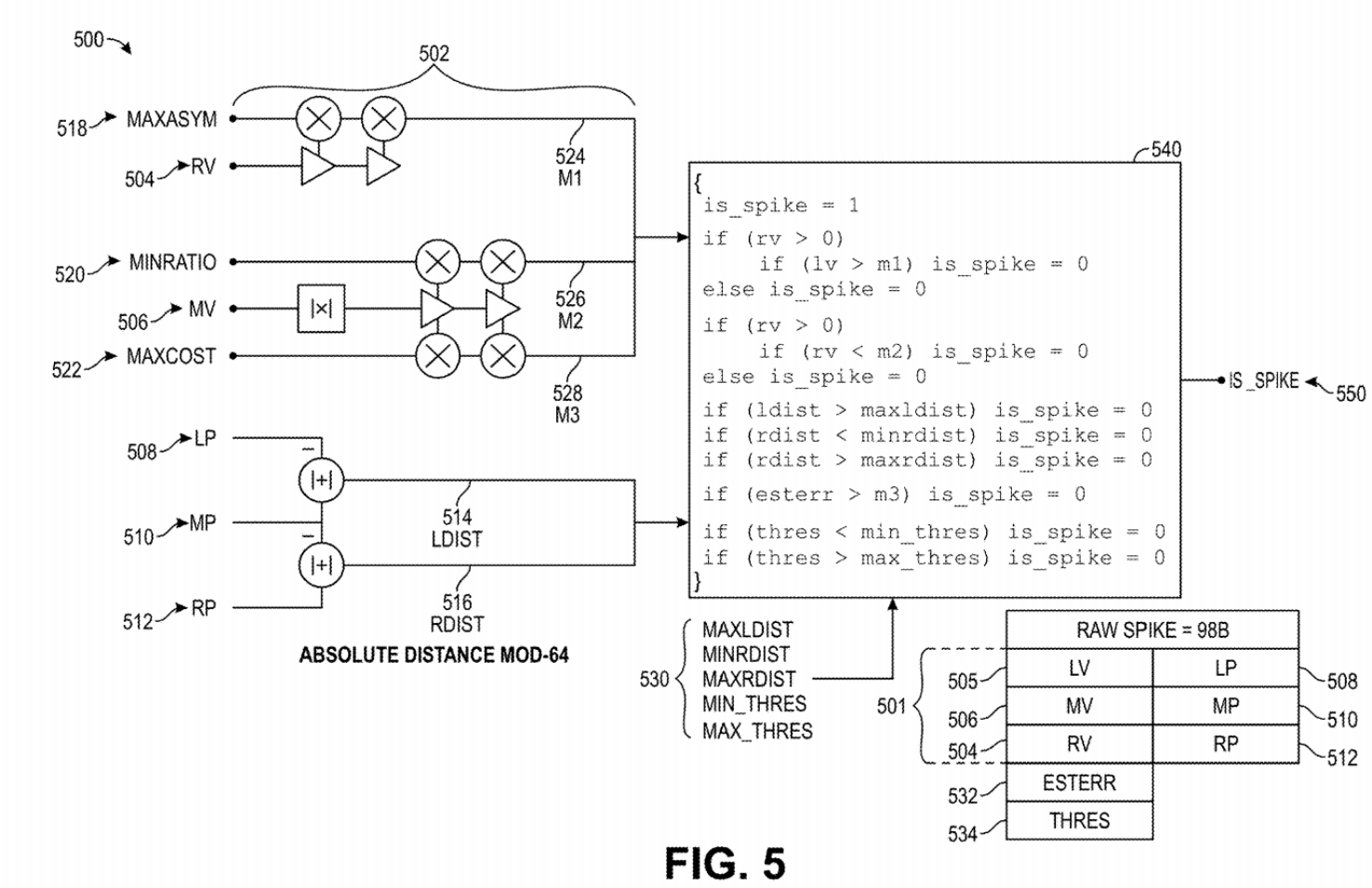
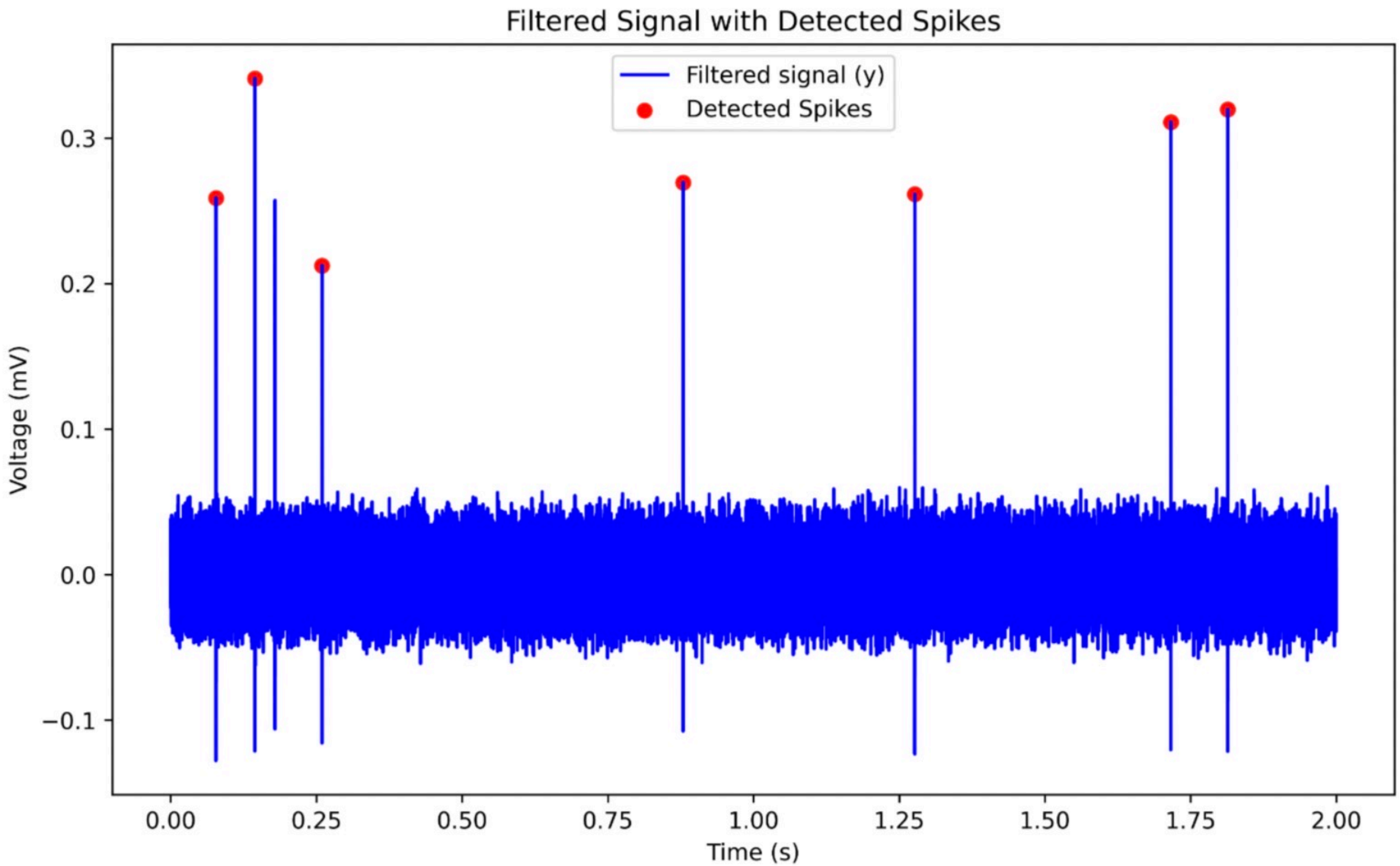


FIG. 5


Filtered Signal with Detected Spikes
Filtered signal (y)
Detected Spikes
Voltage (mV)
0.3
0.2
0.1
0.0
−0.1
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Time (s)

8. Transmit Indication of Spikes

- Transmit **‘1’ + timestamp of when spike is detected (in binary or hex)**
- Can be transmitted via wired or wireless connection

```
bitstream to transmit: 13d9f212d, 13e13f7cf, 13e84f766, 13f6119ce, 13fa36e2f, 13fdbaace, 13fe82c3d,
```

9. Characterizing spikes into bins (additional)

- rdist: time between max and min voltage value
- Can use **ANY metric** to classify (e.g., rdist, ldist, lv, rv, etc.)
- Bins - **Saves memory & time resources** (compared to entire signal)
- In practice: **thousands/millions** of neural signals + spikes!

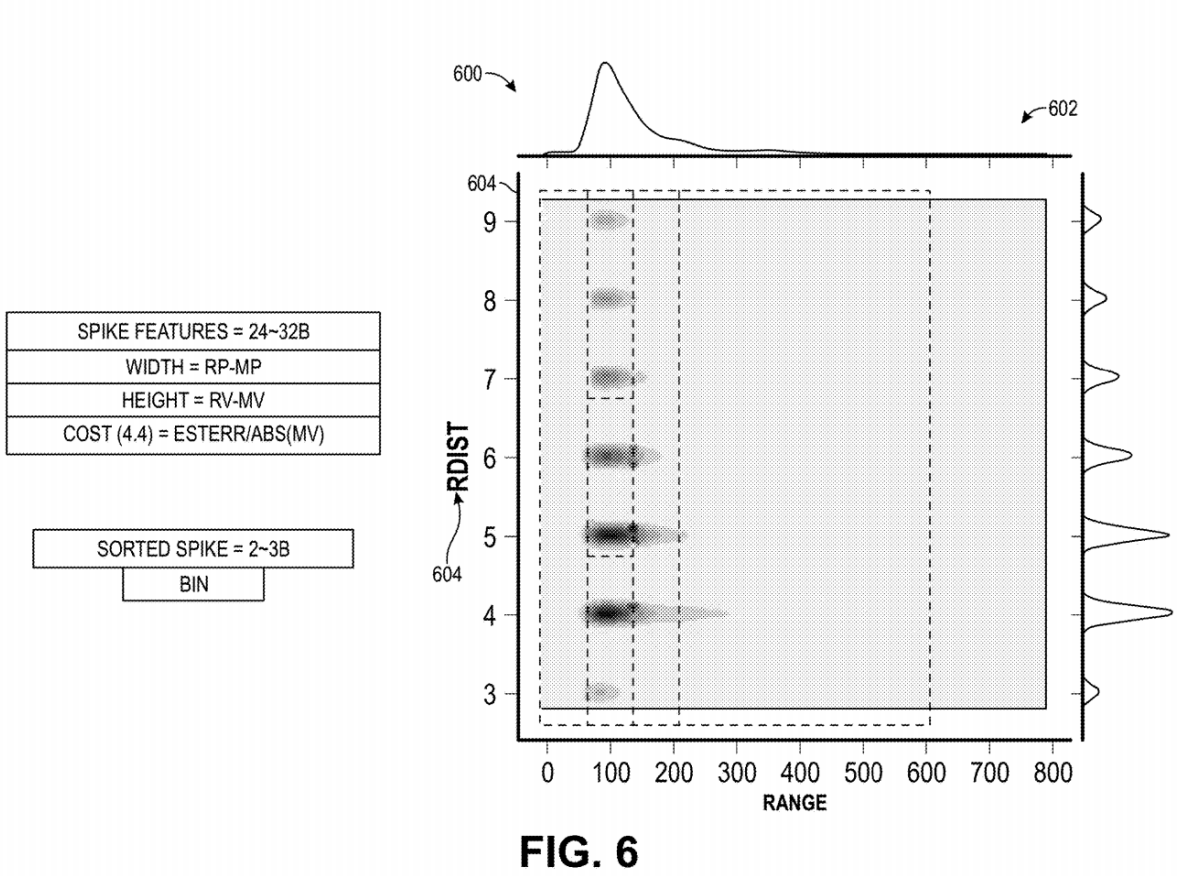
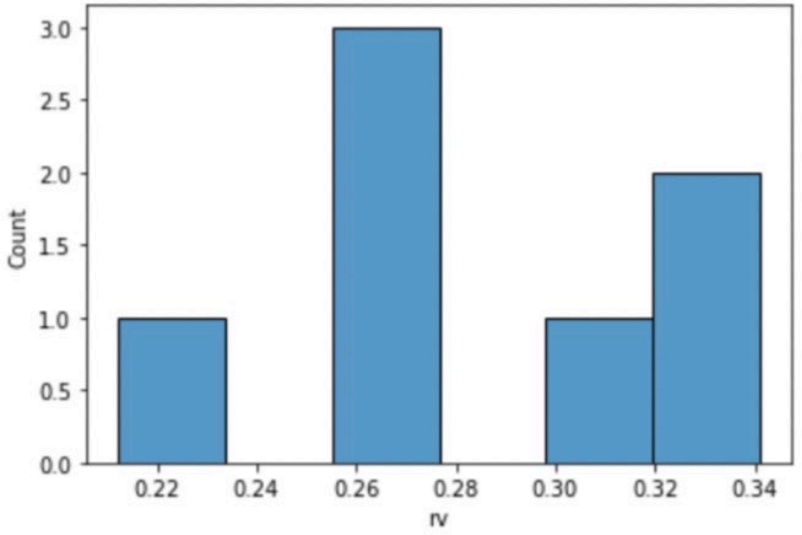


FIG. 6

Example: Spike Characterization based on Right Voltage Value (rv)

- Spikes divided into **6 bins** based on lv (maximum voltage values)
- **Each bin** has a **specific range** and **assigned a value** (**0, 1, 2, 3, 4, 5**)
- More examples shown in notebook

	timestamp of spike (s)	rv	bin
0	0.07770	0.258795	2
1	0.14450	0.340952	5
2	0.25970	0.212276	0
3	0.87930	0.269424	2
4	1.27680	0.261339	2
5	1.71615	0.311008	4
6	1.81385	0.319611	5



Advantages of this system

- **Significant reduction in size** of signal (while **keeping useful information**)
- **Reduce received waveforms** by **3 orders of magnitude** for **low-power wireless transmission**
  - 1000 channels - Gb / s to Mb / s
- Can be performed **much faster** with **lower memory & storage** requirements
- **Near real-time** detection ($\sim \mathbf{1}\boldsymbol{\mu s}$ from receiving signal to spike detection)
- **Less power consumption**
  - 1000 channels - consumes 1-2 mW per channel or less
  - 256 channels - under 50mW (10x more reduction compared to prior systems)

Code Uploaded on Github

- https://github.com/michaela10c/neural_spike_detection/