Manual Threshold selection

Dual-threshold spike detection in one ICA component

By: Javiera Rojas

Improvements

BASE CODE

Manually evaluate independent components (ICs) obtained from ICA and select a threshold to detect spike-like events.

POSSIBILITY TO GO BACK AND REPEAT THE PREVIOUS GIC

MANUALLY ASSIGN A SECOND THRESHOLD TO THE SAME GIC

Inputs and Outputs

INPUTS

possible_gics	Vector of indices indicating ICs to be evaluated.
components	Matrix where each row corresponds to the signal of an IC.

INSIDE CODE INPUTS (USER INPUTS)

continue_input	Repeat previous GIC (0), continue (1), add second threshold (2), or end analysis (9).
user_input	Threshold value to detect peaks in the current GIC. If 0, the GIC is omitted.

OUTPUTS

binary_matrix	Matrix where each row is a binary spike train (1 = spike, 0 = no spike) for a selected GIC.
temp_good_gics	List of selected component indices.
thresh	List of the thresholds used for each selected GIC.

Variables

good_gics	Copy of possible_gics; the list of candidate components to be evaluated manually.
gic_comp	Copy of components; holds the IC signals. Each row is a time series for one IC.
temp_good_gics	Indices of components that were selected as valid (i.e., good GICs) after thresholding.
temp_good_gics_size	Number of good GICs accepted so far. Helps track the growing list temp_good_gics.
thresh	Threshold values (one per good GIC), manually selected by the user to detect spikes.
binary_matrix	Stores the binary spike trains. Each row corresponds to a selected GIC, with 1 where a spike was detected and 0 elsewhere.
i	Current index of the GIC being evaluated in the loop.

Code

Initialization

```
% Close all figures and clear variables
close all
clc
clear

% Load data if not already loaded
load('Mar0916_1_4min_components.mat', 'possible_gics', 'components');

% Copy data into working variables
good_gics = possible_gics;
gic_comp = components;

% Initialize counters and matrix
temp_good_gics_size = 0;
binary_matrix = [];
continue_input = 0;

i = 1;
while i <= length(good_gics)
```

Code

Part 0: User inputs

```
if i > 1 && ~(continue_input == 2)
    if i == length(good_gics)
        continue_input = input(['\n To repeat previous GIC (# ' int2str(i-1) '), press 0.' ...
        '\n To add another (lower) threshold in GIC (# ' int2str(i-1) '), press 2. ' ...
        '\n To end the analysis, press 9:']);
    else
        continue_input = input(['\n To repeat previous GIC (# ' int2str(i-1) '), press 0.' ...
        '\n To continue with GIC (# ' int2str(i) '), press 1. ' ...
        '\n To add another (lower) threshold in GIC (# ' int2str(i-1) '), press 2: ']);
    end

    if continue_input == 0
        i = i - 1;
        disp(['Repeating GIC ' int2str(i)]);
        if ~isempty(binary_matrix)
            temp_good_gics(end) = [];
            thresh(end) = [];
            temp_good_gics_size = temp_good_gics_size - 1;
            binary_matrix(end, :) = [];
        end
        continue_input = 2;
        continue;

    elseif continue_input == 2
        i = i - 1;
        disp(['Adding a threshold in GIC ' int2str(i)]);
        continue;

    elseif continue_input == 9
        return
    end
end
close all;
```

0 → To repeat the threshold selection

1 → To go to the next GIC

2 → To **add** another **LOWER** threshold

9 → To end analysis

If threshold is repeated, goes back and delete the previous threshold and peak detected

Code

Part 1: Manual Threshold Selection

```
fprintf('\n --- Threshold Selection for GIC #%d ---\n', i);

% Plot the component trace
figure(1);
clf;
hold on;
plot(gic_comp(good_gics(i), :));
title(['Component #: ' int2str(good_gics(i))]);
xlabel('x');
ylabel('Amplitude');
pause(0.1);

% Ask user for threshold
user_input = input(['To omit (GIC #' int2str(i) '), press 0. Otherwise, enter threshold for peak detection: ']);

% If user chooses to omit the component
if user_input == 0
    disp(['GIC ' int2str(i) ' omitted.']);
    i = i + 1;
    continue;
end

% Save accepted threshold and GIC index
temp_good_gics_size = temp_good_gics_size + 1;
temp_good_gics(temp_good_gics_size) = good_gics(i);
thresh(temp_good_gics_size) = user_input;
```

Plot the trace to see the best posible threshold

0 → To **omit** the GIC

Other number → **Threshold** selected

Code

Part 2: Peak Detection

```
disp('--- Peak Detection in Progress ---');

% Retrieve trace and threshold
trace = gic_comp(good_gics(i), :);
threshold = user_input;

% Apply peak detection
% If this is a second (lower) threshold for the same GIC,
% mask out spikes already detected by the first threshold
if continue_input == 2
    previous_binary_trace = binary_matrix(end, :);
    binary_trace = get_binary_trace(trace, threshold, previous_binary_trace);
else
    % If this is the first threshold applied to the GIC,
    % detect all peaks above the threshold without masking
    binary_trace = get_binary_trace(trace, threshold);
end

% Save result in binary matrix
binary_matrix = [binary_matrix; binary_trace];
```

If is a second threshold, use the previos binary trace to prevent repeting the same spikes with this lower threshold

Function that returns a binary spike train from a signal trace using threshold-based peak detection (annex 1)

Code

Part 3: Visualization

```
figure(1);
clf;
subplot(2,1,1);
plot(trace);
hold on;
th = yline(threshold, '--r');
legend(th, 'Threshold', 'Location', 'best')
title(['Original Component Trace (Good GIC #' int2str(i) ')']);
xlabel('x');
ylabel('Amplitude');
```

Plot the trace and the threshold selected

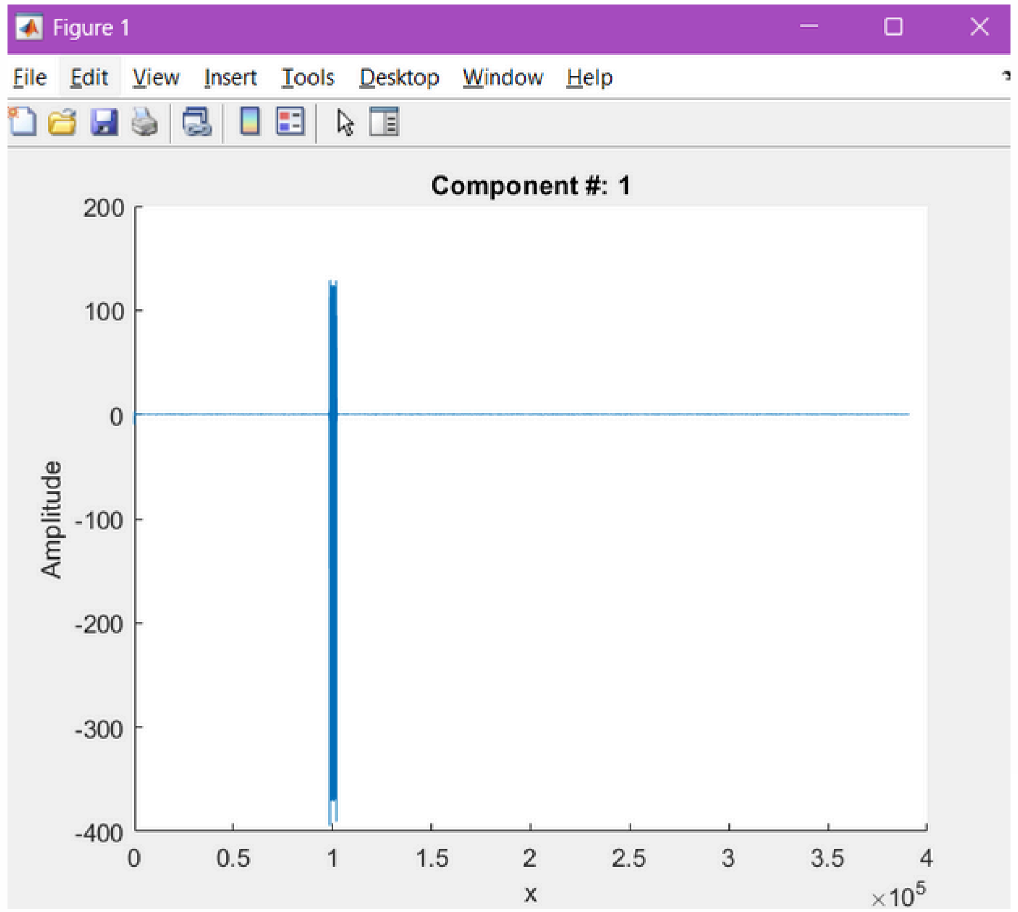
```
subplot(2,1,2);
plot(binary_trace);
title('Binary Spike Train (Detected Peaks)');
xlabel('x');
ylabel('1 → spike');
pause(0.1);
```

Spikes detected with the threshold selection

```
i = i + 1;
continue_input = 0;
```

Example

GIC 1 (omit)



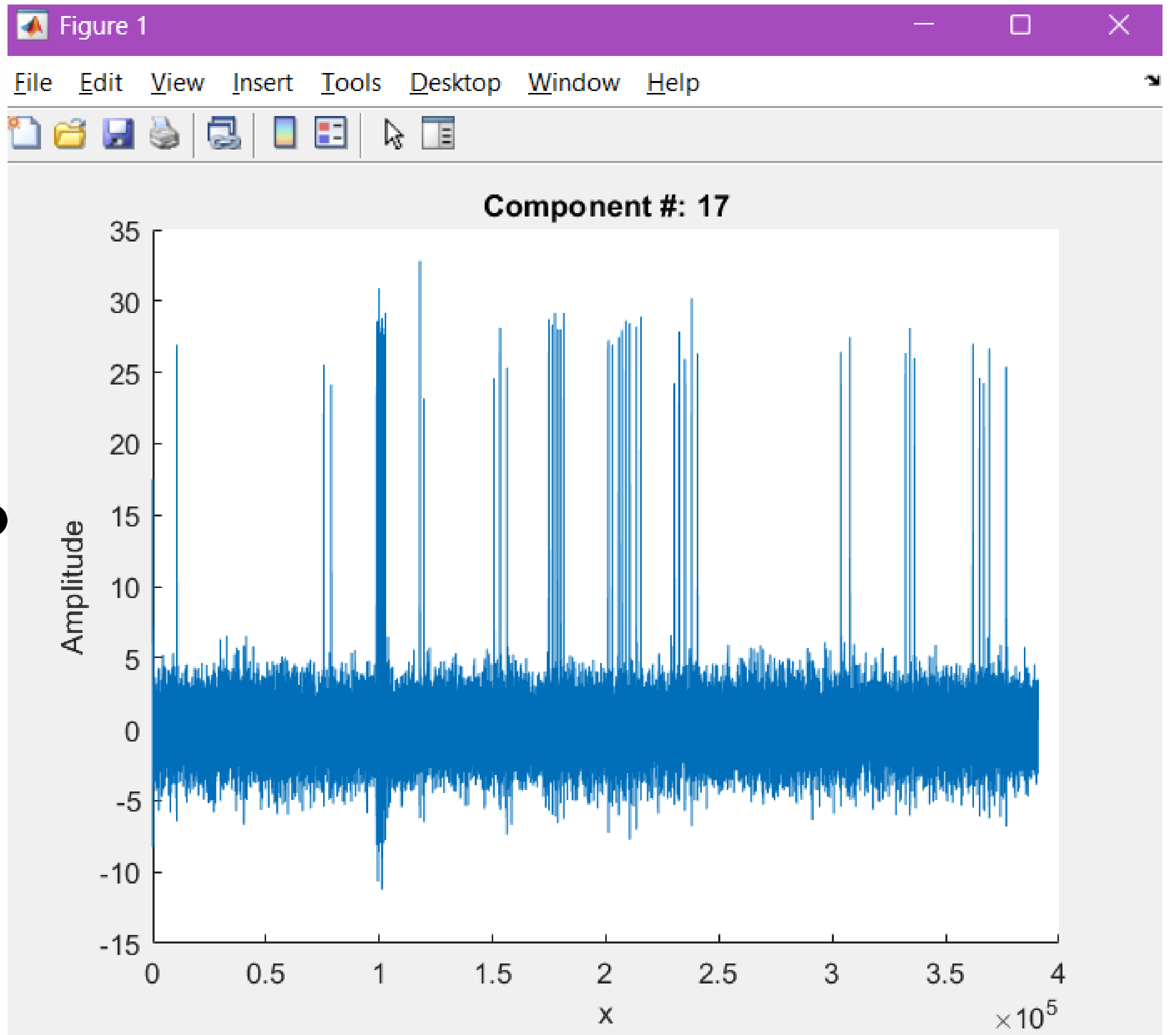

```
--- Threshold Selection for GIC #1 ---
To omit (GIC #1), press 0. Otherwise, enter threshold for peak detection:
```

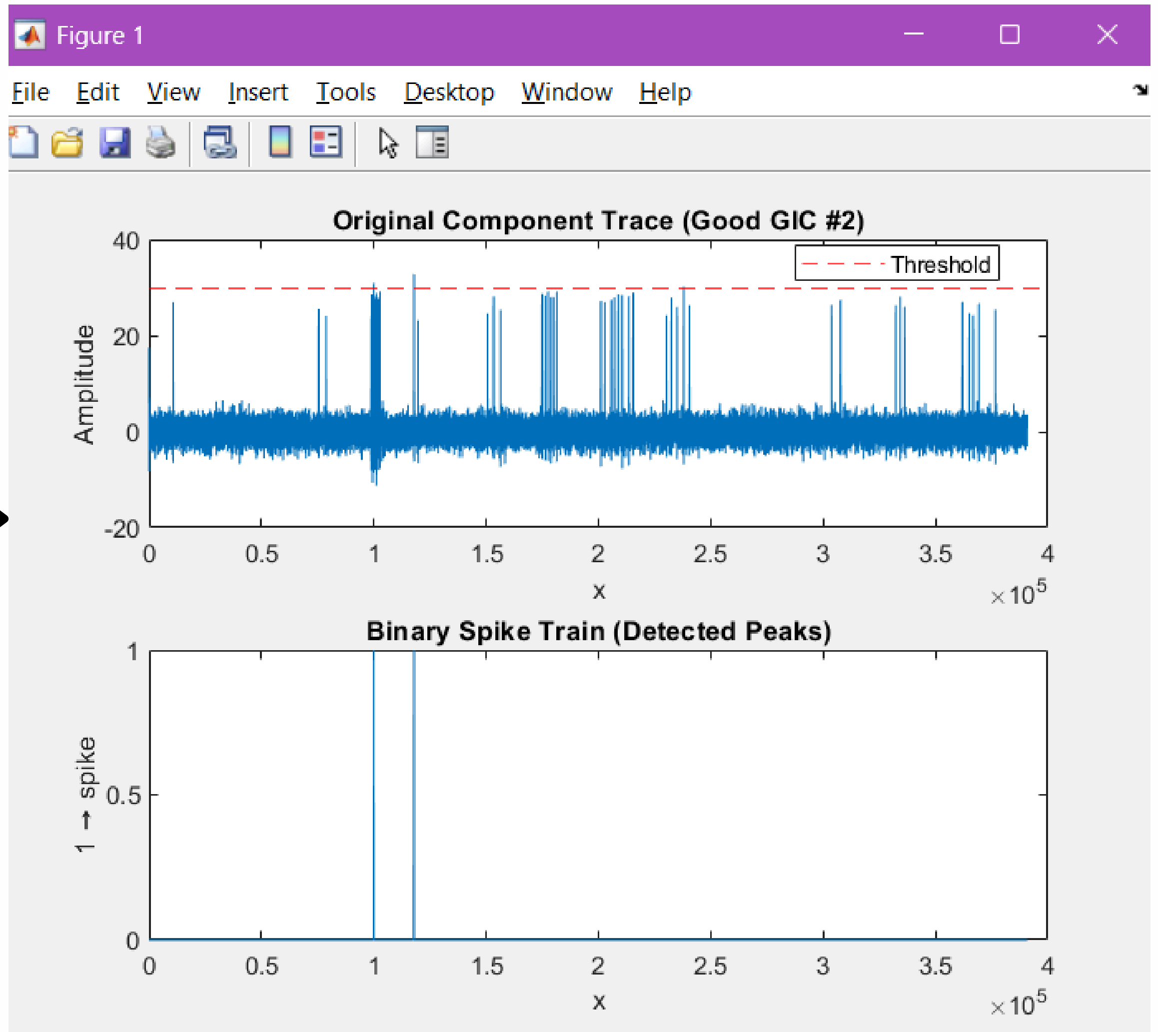
0 → OMIT GIC

```
 --- Threshold Selection for GIC #1 ---
To omit (GIC #1), press 0. Otherwise, enter threshold for peak detection: 0
GIC 1 omitted.

 To repeat previous GIC (# 1), press 0.
 To continue with GIC (# 2), press 1.
 To add another (lower) threshold in GIC (# 1), press 2:
```

Example

GIC 2 (repeat)



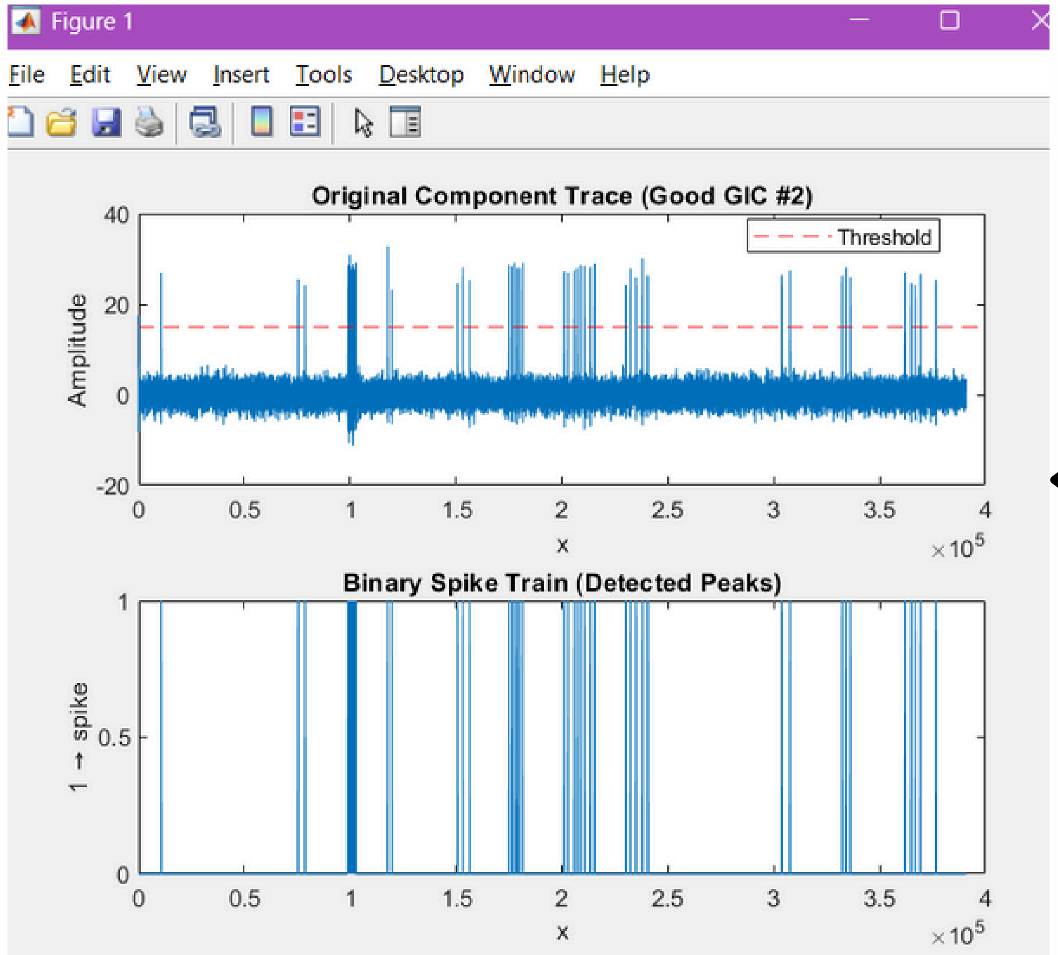



```
--- Threshold Selection for GIC #2 ---
To omit (GIC #2), press 0. Otherwise, enter threshold for peak detection: 30
--- Peak Detection in Progress ---
```

Example
GIC 2 (repeat)

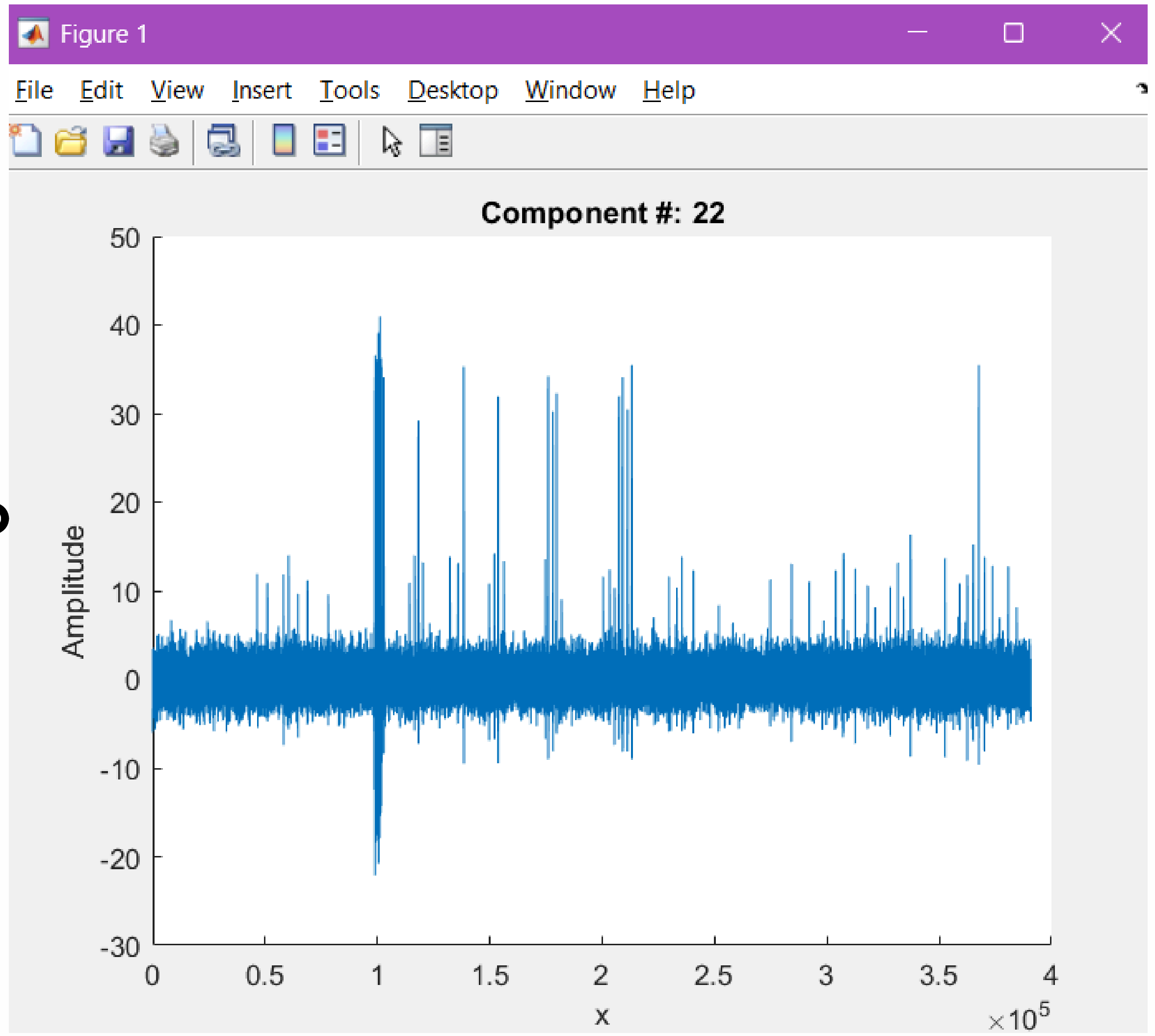
```
 --- Threshold Selection for GIC #2 ---
To omit (GIC #2), press 0. Otherwise, enter threshold for peak detection: 30
--- Peak Detection in Progress ---

 To repeat previous GIC (# 2), press 0.
 To continue with GIC (# 3), press 1.
 To add another (lower) threshold in GIC (# 2), press 2: 0
Repeating GIC 2

 --- Threshold Selection for GIC #2 ---
To omit (GIC #2), press 0. Otherwise, enter threshold for peak detection: 15
--- Peak Detection in Progress ---
```

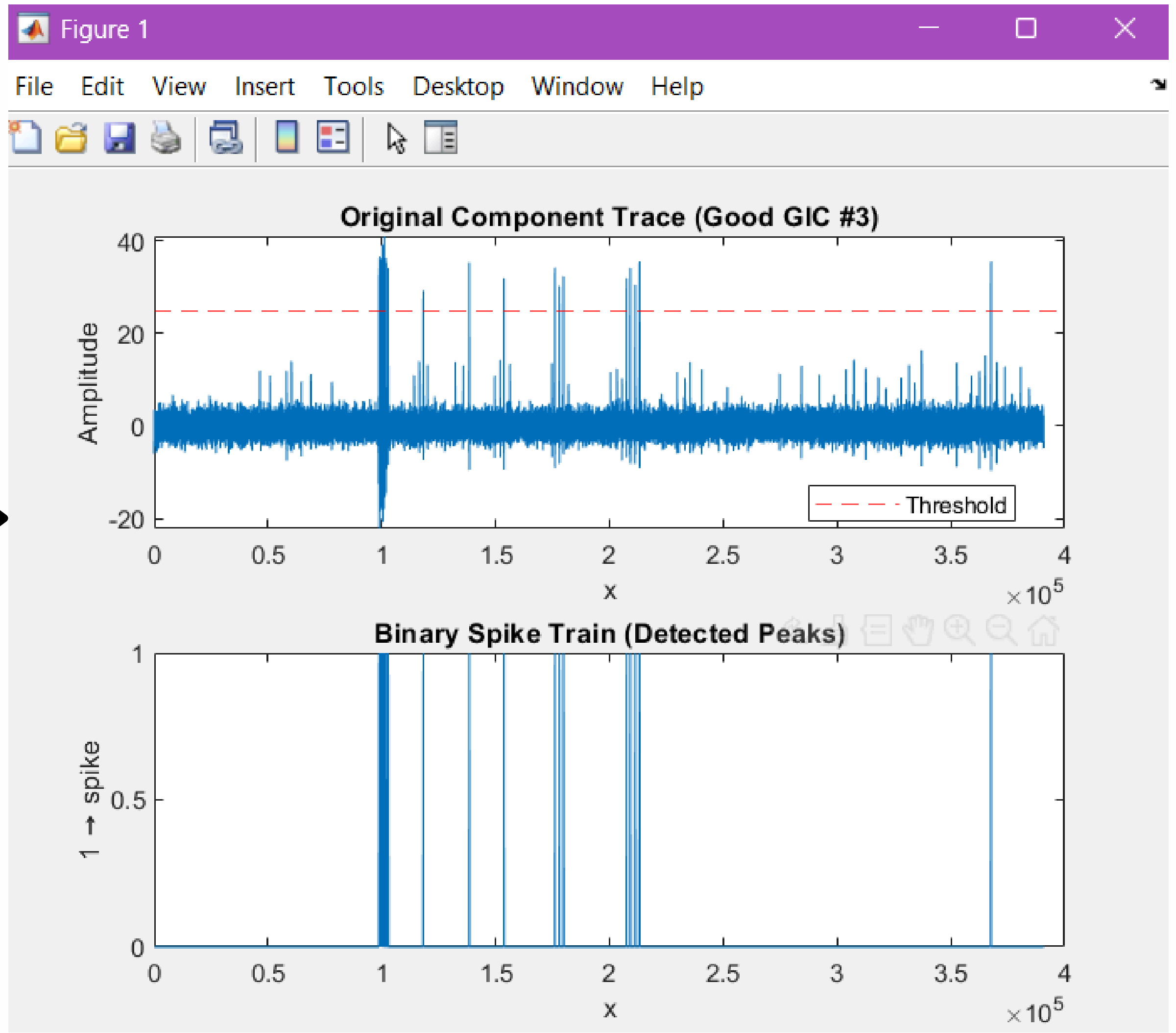
0 → REPEAT GIC



Example

GIC 3 (Add a 2nd threshold)






```
To repeat previous GIC (# 2), press 0.
To continue with GIC (# 3), press 1.
To add another (lower) threshold in GIC (# 2), press 2: 1

--- Threshold Selection for GIC #3 ---
To omit (GIC #3), press 0. Otherwise, enter threshold for peak detection: 25
--- Peak Detection in Progress ---
```

1 → NEXT GIC

1ST THRESHOLD (HIGER)

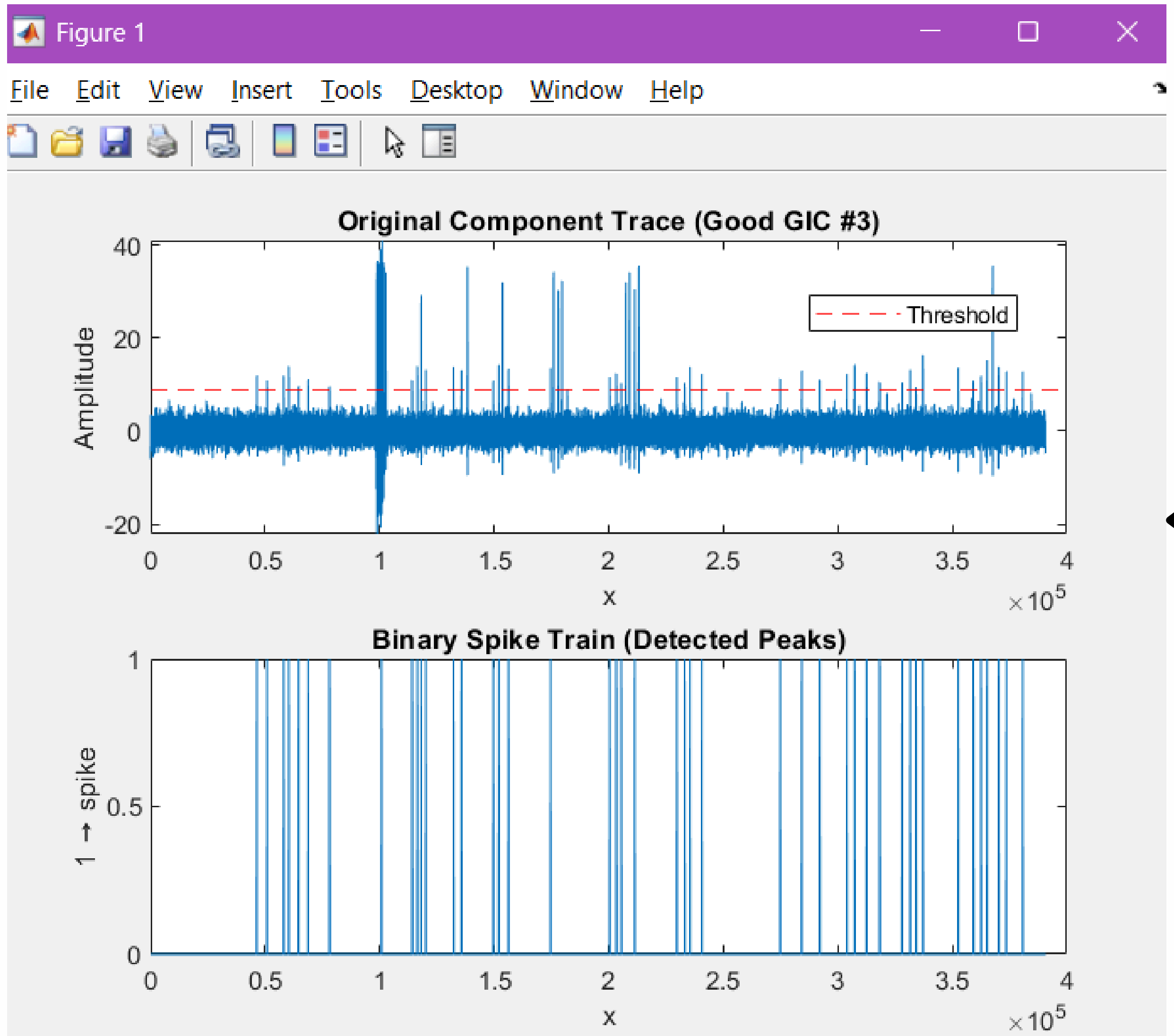
Example

GIC 3 (Add a 2nd threshold)

```
 To repeat previous GIC (# 3), press 0.
 To continue with GIC (# 4), press 1.
 To add another (lower) threshold in GIC (# 3), press 2: 2
Adding a threshold in GIC 3

 --- Threshold Selection for GIC #3 ---
To omit (GIC #3), press 0. Otherwise, enter threshold for peak detection: 9
--- Peak Detection in Progress ---
```

2 → ADD A THRESHOLD

2ST THRESHOLD (LOWER)



Manual Threshold selection

Dual-threshold spike detection in one ICA component

Function
get_binary_trace

```matlab
function binary_trace = get_binary_trace(trace, threshold, mask_trace)
    % Returns a binary spike train from a signal trace using threshold-based peak detection
    %   INPUTS:
    %       trace       : Vector representing the signal trace of an ICA component.
    %       threshold   : Scalar value. The user-defined threshold for detecting peaks.
    %
    %   OUTPUT:
    %       binary_trace: Binary vector where each entry is:
    %           - 1 if a spike (peak) is detected
    %           - 0 otherwise

    n = size(trace,2);
    trace1 = (trace > threshold);               % 1 where trace is above threshold, 0 otherwise
    trace_high = (trace - threshold).* trace1;  % Keep only the values above the threshold

    % Operation matrix
        % Dimensions: 8 rows (operations), (n + 2) columns for safe indexing
    new_n = n + 2;
    traceop_mat = zeros(8, new_n);

    % Row 1: Thresholded signal
    traceop_mat(1, 2:new_n-1)   = trace_high;

    % Row 2: Lagged version of Row 1 (previous time step) (column 1 moves 1 up)
    traceop_mat(2, 1:new_n-2)   = traceop_mat(1, 2:new_n-1);

    % Row 3: First difference (discrete derivative)
    traceop_mat(3, :)           = traceop_mat(1,:) - traceop_mat(2,:);

    % Row 4: Keep only rising edges (positive threshold crossings)
        % (traceop_mat(1,:) > 0) -> 1 if is true, 0 if not
    traceop_mat(4, :)           = traceop_mat(3,:).*(traceop_mat(1,:) > 0);

    % Row 5: Lagged version of Row 4 (previous derivative) (column 4 moves 1 down)
    traceop_mat(5, 2:new_n-1)   = traceop_mat(4, 1:new_n-2);

    % Row 6:  Ratio between current and previous derivative (slope ratio)
        % Detects abrupts changes
    traceop_mat(6, :)           = traceop_mat(4, :) ./ traceop_mat(5, :);

    % Row 7:  Remove inf values
        % ~isinf(traceop_mat(6,:)) -> 1 if is NOT inf, 0 if it inf
    traceop_mat(7, :)           =  (~isinf(traceop_mat(6, :))) .* traceop_mat(6, :);

    % Row 8: Detect negative slope transitions (Peak detection)
        % traceop_mat(7,:) < 0 -> 1 if its an abrupt change (peak), 0 if its not
    traceop_mat(8, :)           = traceop_mat(7, :) < 0;

    % Peak matrix
    binary_trace = traceop_mat(8, 2:n-1);

    % If a masking binary trace is provided (e.g., from a previous higher threshold),
        % Exclude already-detected spikes. This is useful when applying a second, lower
        % threshold to the same trace in order to isolate smaller spikes from a second neuron.
    if nargin == 3 && ~isempty(mask_trace)
        binary_trace = binary_trace & ~mask_trace;
    end

end
```