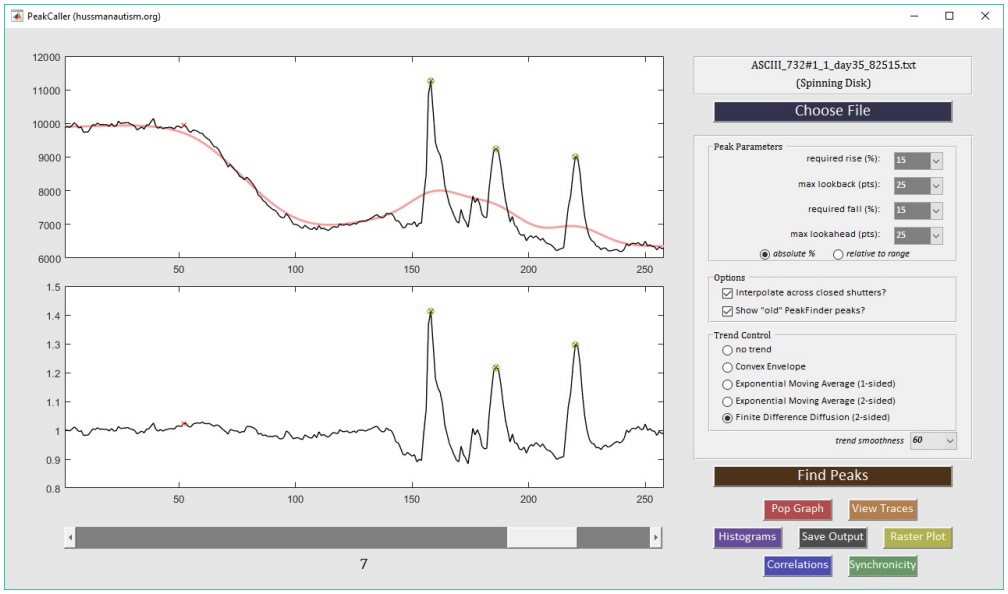
1 PeakCaller Primer



After climbing a great hill, one only finds that there are many more hills to climb.

- Nelson Mandela

Introduction

Calcium imaging provides scientists with a window to see into the activity of neurons and neural networks. Imaging using modern microscopes (like *spinning disk* or *array scan* confocals) can generate huge amounts of data about changing calcium concentrations within cells – and a trained eye can identify regions of interest within a sample and can recognize *spikes* or *peaks* in noisy data. Our goal is to provide a utility that will *automate* the process of identifying these peaks within regions of interest, in order to make measuring neuronal activity both faster and more robust.

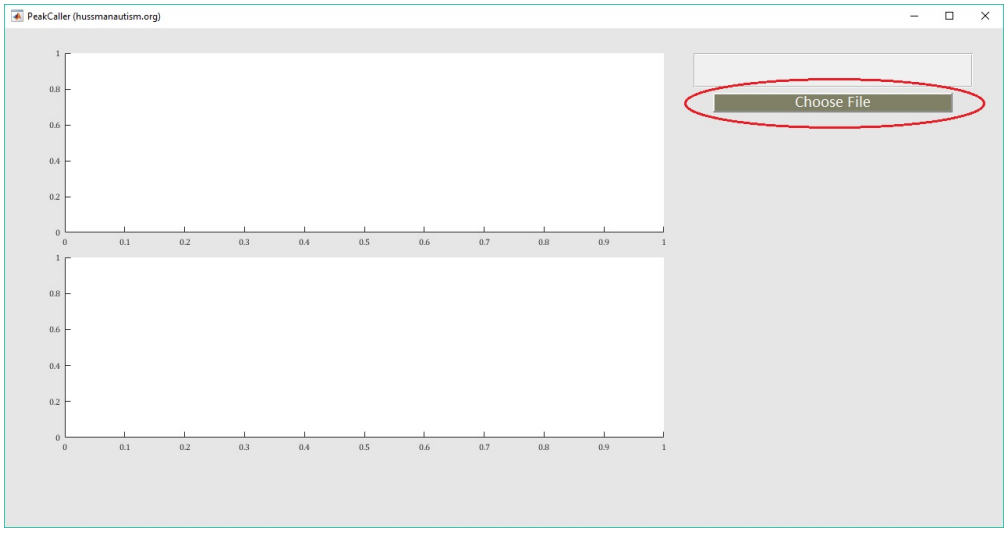


Figure 1.1: On startup, the user has no option but to choose a data file.

PeakCaller is the MATLAB software that we have developed for this purpose. This primer is meant to serve as an introduction to that software. In the following sections, we will describe how a user can interact with **PeakCaller**, and we will detail all of the computations that **PeakCaller** performs.

1.1 Using PeakCaller

PeakCaller is designed to be easy to run, to provide a simple way to visualize results, and to record those results in a concise and convenient format.

Starting up **PeakCaller**, the user is presented with two empty sets of axes on the left and a single button labeled *Choose File* on the right, see Figure 1.1. Clicking this button opens up a standard dialog box that prompts the user to pick a data file that they would like to analyze, as in Figure 1.2.

PeakCaller can recognize data from either *spinning disk* or *array scan* confocal microscopes, saved as comma-delimited files. These two different systems produce data in two different formats, shown in Figure 1.3. Note

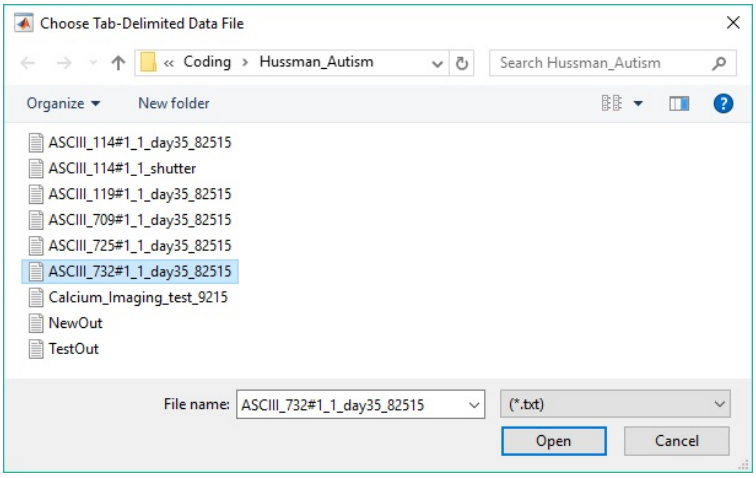


Figure 1.2: The user chooses a data file using a standard dialog box.

that for a spinning disk confocal, there are time stamps in the first column (starting in the third row), and the data for the various regions of interest runs vertically down the remaining columns. For the array scan confocal, time is not explicitly given, although there is an index in the first row (starting in the fourth column). The data for the various regions runs horizontally across the remaining columns.

PeakCaller differentiates between these two different types of data files in a fairly crude way. It attempts to grab a huge rectangular array of numbers that has the form of the spinning disk data – with its upper left-hand corner in the third row of the first column. The first column will be the time stamps and the remaining columns the intensity profiles. If that is not successful (because there are elements in the array that aren't numbers), it then tries to grab an array that has the form of the array scan data – with its upper left-hand corner in the second row of the fourth column. If that doesn't work, the software complains.

Once a data set is successfully loaded, the software displays a button labeled *Find Peaks*, along with a panel that allows the user to fine-tune the qualities they require of peaks in their data. First, a user can choose to *de-trend* the data, removing any long-term changes in level that can disguise the shorter-term behavior. There are several options available for approximating the

	A	B	C	D	E	F
1	Time::Tim	Channel_(	Channel_(	Channel_(	Channel_(	Channel
2	Min					
3	0	12661.09	14933.97	20256.71	20847.28	14029.7
4	0.038752	12686.6	14989.48	20248.21	20765.91	14053.1
5	0.077554	12441.64	14977.22	20070.46	20691.67	14025.3
6	0.116607	12240.91	15026.24	20111.64	20617.64	13966.7
7	0.155509	12117.43	14950.94	20089.27	20635.18	13887.
8	0.194294	12095.07	14798.62	20049.51	20682.91	13835.6
9	0.233063	12009.49	14852.49	20177.44	20697.75	13884.6
10	0.272232	11787.07	15199.01	20012.91	20560.48	13766.3
11	0.311001	12018.08	15244.84	19847.47	20525.79	13819.4
12	0.349887	12191.03	15005.15	19902.65	20381.62	13867.8
13	0.388656	12186.53	14769.34	19807.97	20458.02	13863.3
14	0.427424	12317.05	14607.03	19792.45	20520.73	13899.7

	A	B	C	D	E	F	G
1	Well	Field	Cell	t1	t2	t3	t4	t5
2	A2	1	1	23398	23804	24374	23286	
3	A2	1	2	27	60	109	33	
4	A2	1	3	49330	44587	47775	45395	
5	A2	1	4	12614	12211	12717	11870	
6	A2	1	5	13131	13816	13824	14622	
7	A2	1	6	7513	8392	7562	9185	
8	A2	1	7	9492	7574	8335	8020	
9	A2	1	8	60911	63549	64437	67784	
10	A2	1	9	42731	44595	46159	47630	
11	A2	1	10	22559	70475	77501	27314	
12	A2	1	11	54942	57534	60849	61571	
13	A2	1	12	22172	22360	21960	23001	
14	A2	1	13	60089	11547	9621	60365	

Figure 1.3: **PeakCaller** can accept data from either spinning disk or array scan confocal microscopes. The left-hand panel in this figure shows data produced by a spinning disk confocal microscope. The right-hand panel is from an array scan confocal microscope.

data's long-term trend; these are detailed in Section 1.2. Second, the user can *define* the characteristics that a "peak" should have: how far and how fast should the data rise – and fall – for a point to be labeled as a peak? The parameters governing these characteristics are detailed in Section 1.3.

Once the user is happy with their answers for what makes a peak, they simply press the *Find Peaks* button and let the software go to work. The results are displayed visually on the two left-hand graphs – the upper graph shows the original data and the defined long-term trend, while the lower graph shows the *de-trended* data. Each has all identified peaks marked with small green circles. A scroll bar below the graphs allows the user to quickly flip through the results for *each* of the different regions of interest included in the data file.

Finally, six buttons on the right-hand side offer the user options for processing and understanding the results. Choosing *Pop Graph* opens the lower graph in a new window, allowing the user to have a copy of the graph that is easier to manipulate and edit. Choosing *View Traces* allows the user to see *all* of the traces in a single plot, with the peaks highlighted in a different color. *Raster Plot* provides another view of the locations of the peaks across the various ROIs. *Histograms* opens a figure summarizing the properties of the peaks within the file. The figure contains two histograms: one collecting

the heights of the peaks and one collecting the rise and fall times. *Correlations* moves pairwise through all of the fluorescence traces, computing the cross-correlation and summarizing the results in a heat map. *Synchronicity* displays a synchronization matrix built using only information about the peaks, rather than the entire fluorescence traces. Finally, and unsurprisingly, choosing *Save Output* allows the user to export information about the peaks identified by **PeakCaller** to a file. For each individual peak, the software records measures of peak location, normalized peak height, onset time and decay time. For each individual ROI, it also provides a summary of the mean characteristics of all the peaks identified for that ROI. And for the entire collection of ROIs, it provides a summary pulling together the mean characteristics all of the peaks in the entire loaded data file. This output is saved as three sheets of an Excel file. If the user chooses to write the data to a file that already exists, **PeakCaller** will delete and replace this existing file.

1.2 Trend Control

PeakCaller offers several tunable options for controlling for the *trend* of a time series. The user is encouraged to experiment with the different choices to determine which option produces the best results in their samples.

Finite Difference Diffusion

Have you ever popped a helium balloon at a birthday party? Where does the helium go after it is released from the balloon? Well, at the moment the balloon pops, the helium is localized in the area where the balloon was. But as time goes on, the helium atoms bounce and scatter, slowly taking on a more uniform distribution in the party room. A similar situation to consider is an ice cube dropped into a hot beverage. First, the parts of the beverage closest to the ice cube cool off, and as time goes by the temperature profile of the beverage settles into a uniform lukewarm state.

Finite difference diffusion numerically approximates this behavior (modelled by a partial differential equation called the *heat equation* or the *diffusion equation*) in order to smooth each time series and approximate the long-term trend. The parameter *trend smoothness* governs how long this

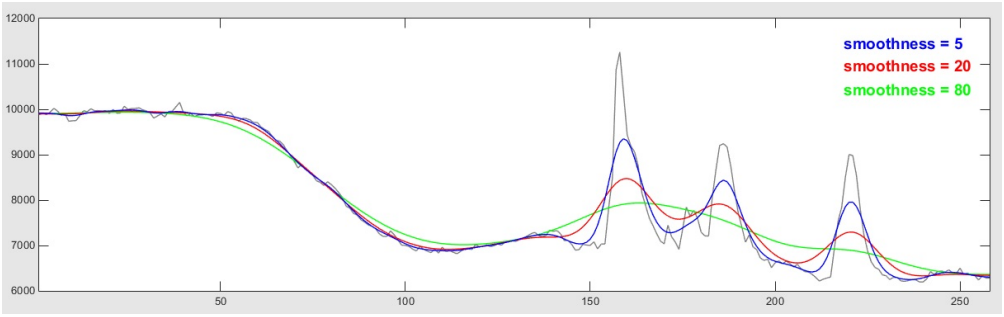


Figure 1.4: In grey, we see an example calcium profile. Time is on the horizontal axis and the intensity of a calcium-binding dye is on the vertical axis. The long-term trend (modelled using *finite difference diffusion*) is also shown, for three different values of the *trend smoothness* parameter. The larger this parameter gets, the more the fine structure of the profile melts away....

diffusion process is allowed to take place. A small number represents a short amount of time (the balloon was just popped or the ice cube was just added to the beverage) and a large number represents a longer amount of time (the helium atoms have had more time to scatter around the room, or the temperature has had more time to flow throughout the beverage). Figure 1.4 shows an example calcium profile, along with several *smoothings*, for increasing values of the *trend smoothness* parameter.

For the numerically curious, suppose that the original example profile is defined by the equally-spaced values $\{x_0^1, x_0^2, \ldots, x_0^T\}$. (Note that the superscripts here are simply meant to index the measurements at different times and are *not* meant to indicate exponentiation.) We *melt* this profile in quarter steps according to the recursion

$$x_{n+1}^t = x_n^t + \frac{x_n^{t-1} - 2x_n^t + x_n^{t+1}}{4}$$

for $2 \leq t \leq T-1$. Computationally, we must take special care at the beginning and end of the data set, that is, when $t = 1$ and $t = T$. We model the ends using *reflecting boundaries*, and thus when $t = 1$ and $t = T$

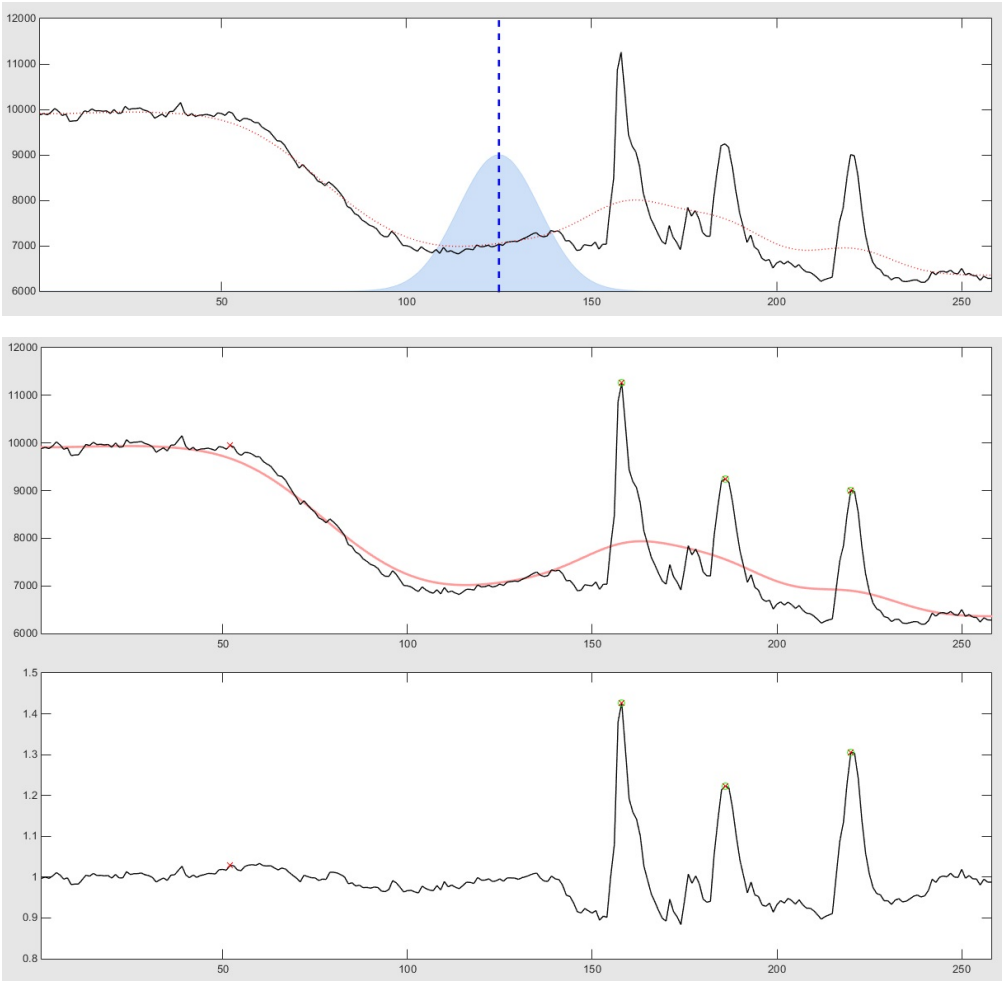


Figure 1.5: The top panel in this figure illustrates the weight function corresponding to *finite difference diffusion* with the *trend smoothness* parameter set to 80, and is approximately *Gaussian*. For the original profile (in black), the *smoothing* at the dashed blue line is computed by weighting the values of the original profile proportionally with the *weight function* shaded in light blue. The lower two panels show the actual graphics provided by **PeakCaller**. The middle graph shows the original profile, along with the smoother trend (still using *finite difference diffusion* with the *trend smoothness* parameter set to 80). The lower graph shows the *de-trended* data series, which is simply the quotient of the original data divided by the smoothed data. Peaks are identified using this de-trended data series.

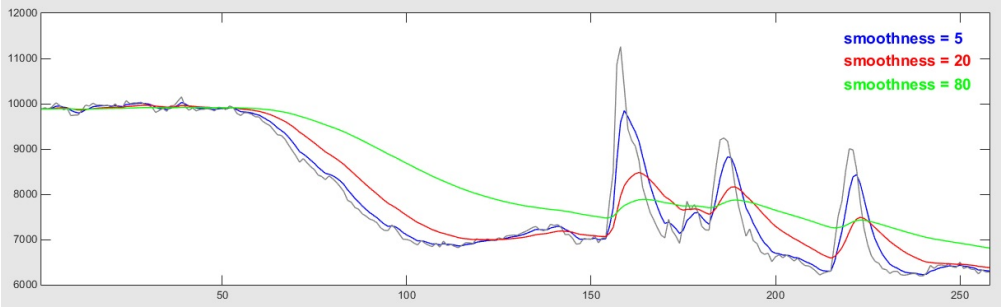


Figure 1.6: In grey, we see an example calcium profile. Time is on the horizontal axis and the intensity of a calcium-binding dye is on the vertical axis. The long-term trend (modelled using *a 1-sided exponential moving average*) is also shown, for three different values of the *trend smoothness* parameter. The larger this parameter gets, the more the fine structure of the profile melts away.

we have the relationships

$$x^1_{n+1} = x^1_n + \frac{2x^2_n - 2x^1_n}{4} \qquad \text{and} \qquad x^T_{n+1} = x^T_n + \frac{2x^{T-1}_n - 2x^T_n}{4}.$$

If the *trend smoothness* parameter is called s, we proceed for a number of steps equal to $4s$, and so the smoothed profile is given by

$$\{x^1_{4s}, x^2_{4s}, \ldots, x^T_{4s}\}.$$

A cartoon showing how the values of the original profile are weighted in order to generate this smoothed profile is displayed in the top panel of Figure 1.5.

Exponential Moving Average (1-sided)

In financial applications, long-term trends in a data series are often defined by considering *simple moving averages* or *exponential moving averages* of that data. Some stock trading methods initiate buys and sells based upon

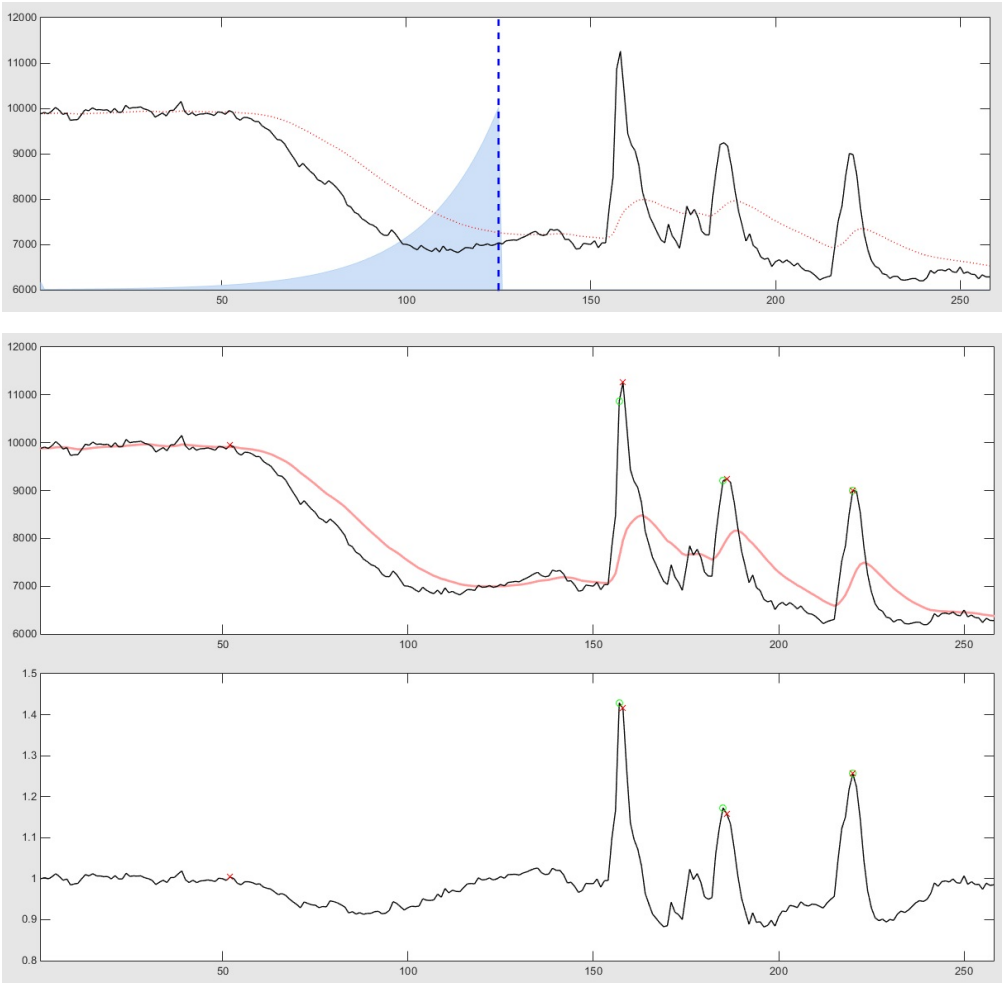


Figure 1.7: The top panel in this figure illustrates the weight function corresponding to *a 1-sided exponential moving average* with the *trend smoothness* parameter set to 40. For the original profile (in black), the *smoothing* at the dashed blue line is computed by weighting the values of the original profile proportionally with the *weight function* shaded in light blue. Notice especially that the smoothed value at any point depends only on the portion of the original profile to the left of that point. The lower two panels show the actual graphics provided by **PeakCaller**. The middle graph shows the original profile, along with the smoothing (still using *a 1-sided exponential moving average*, but here with the *trend smoothness* parameter set to 20). The lower graph shows the *de-trended* data series, which is simply the quotient of the original data divided by the smoothed data. Peaks are identified using this de-trended data series.

a stock's price *breaking through* a long-term trend line, for example, its 200-day moving average. In the study of calcium signals, we can use moving averages to approximate trends in the same way.

The exponential moving average is defined in a single run, from left to right. If the original example profile is defined by the equally-spaced values $\{x_0^1, x_0^2, \ldots, x_0^T\}$, then the smoothed profile is defined recursively by

$$x_s^1 = x_0^1 \qquad \text{and} \qquad x_s^t = \frac{s-1}{s+1} x_s^{t-1} \; + \; \frac{2}{s+1} x_0^t,$$

for $t = 2, 3, \ldots, T$ and where s is the *trend smoothness* parameter. Notice that, in terms of the original example profile, the newly-defined smoothed profile telescopes as

$$x_s^t = \frac{2}{s+1} x_0^t + \frac{2(s-1)}{(s+1)^2} x_0^{t-1} + \frac{2(s-1)^2}{(s+1)^3} x_0^{t-2} + \frac{2(s-1)^3}{(s+1)^4} x_0^{t-3} + \ldots,$$

where the sum continues until it reaches x_0^1. The coefficient of x_0^1 does not follow the pattern of the others, but is *boosted* so that the sum of all of the coefficients is one.

Figure 1.6 shows an example calcium profile, along with several smoothings using *exponential moving averages*, for increasing values of the *trend smoothness* parameter. A cartoon showing how the values of the original profile are weighted in order to generate the smoothed profile is displayed in the top panel of Figure 1.7.

Exponential Moving Average (2-sided)

In financial applications, generally only the past history is known; an estimate of the current trend cannot make use of future data that is not yet in evidence. In the current context however, we already have the entire tape of a neuron's calcium signal; and thus can approximate the trend at any point by using measurements taken both before *and after* that particular moment in time. One option is to simply *average* a backward-looking exponential moving average (looking back to the beginning of the tape) with a forward-looking exponential moving average (looking forward to the end of the tape). This is exactly the computation we make.

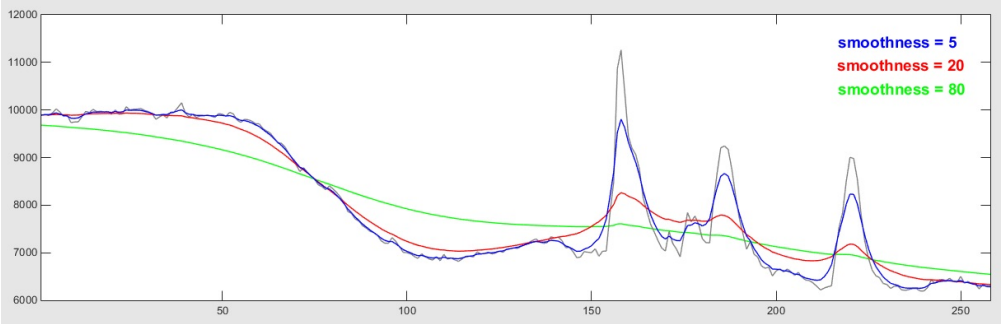


Figure 1.8: In grey, we see an example calcium profile. Time is on the horizontal axis and the intensity of a calcium-binding dye is on the vertical axis. The long-term trend (modelled using *a 1-sided exponential moving average*) is also shown, for three different values of the *trend smoothness* parameter. The larger this parameter gets, the more the fine structure of the profile melts away.

Numerically, we compute the two runs independently, one from left to right and another from right to left. If the original example profile is defined by the equally-spaced values $\{x_0^1, x_0^2, \ldots, x_0^T\}$, then the backward-looking exponential moving average at each point is defined recursively (just as before) by

$$b_s^1 = x_0^1 \qquad \text{and} \qquad b_s^t = \frac{s-1}{s+1} b_s^{t-1} \; + \; \frac{2}{s+1} x_0^t,$$

for $t = 2, 3, \ldots, T$, where s is the *trend smoothness* parameter. The forward-looking exponential moving average is defined similarly, but starts at the end of the interval and works backwards:

$$f_s^T = x_0^T \qquad \text{and} \qquad f_s^t = \frac{s-1}{s+1} f_s^{t+1} \; + \; \frac{2}{s+1} x_0^t,$$

for $t = T-1, T-2, \ldots, 1$. In the end, we combine these forward and backward-looking estimates in order to approximate a two-sided smoothing $\{x_s^1, x_s^2, \ldots, x_s^T\}$, where

$$x_s^t = \frac{b_s^t + f_s^t}{2}$$

for $t = 1, 2, \ldots, T$.

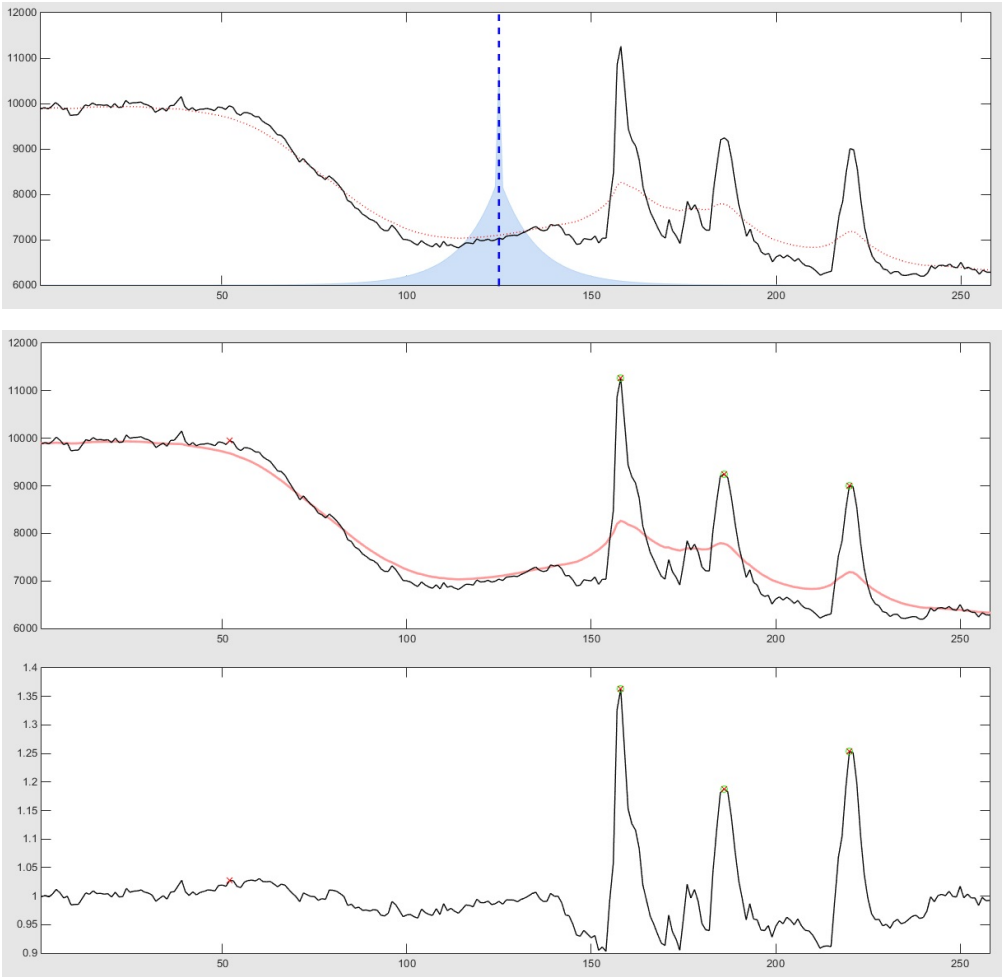


Figure 1.9: The top panel in this figure illustrates the weight function corresponding to *a 2-sided exponential moving average* with the *trend smoothness* parameter set to 20. For the original profile (in black), the *smoothing* at the dashed blue line is computed by weighting the values of the original profile proportionally with the *weight function* shaded in light blue. Notice especially the double-weight given to the value at the blue line, as it is the only point in common to both the forward and backward looking moving averages. The lower two panels show the actual graphics provided by **PeakCaller**. The middle graph shows the original profile, along with the smoothing (*a 2-sided exponential moving average*, with the *trend smoothness* parameter set to 20). The lower graph shows the *de-trended* data series, which is simply the quotient of the original data divided by the smoothed data. Peaks are identified using this de-trended data series.

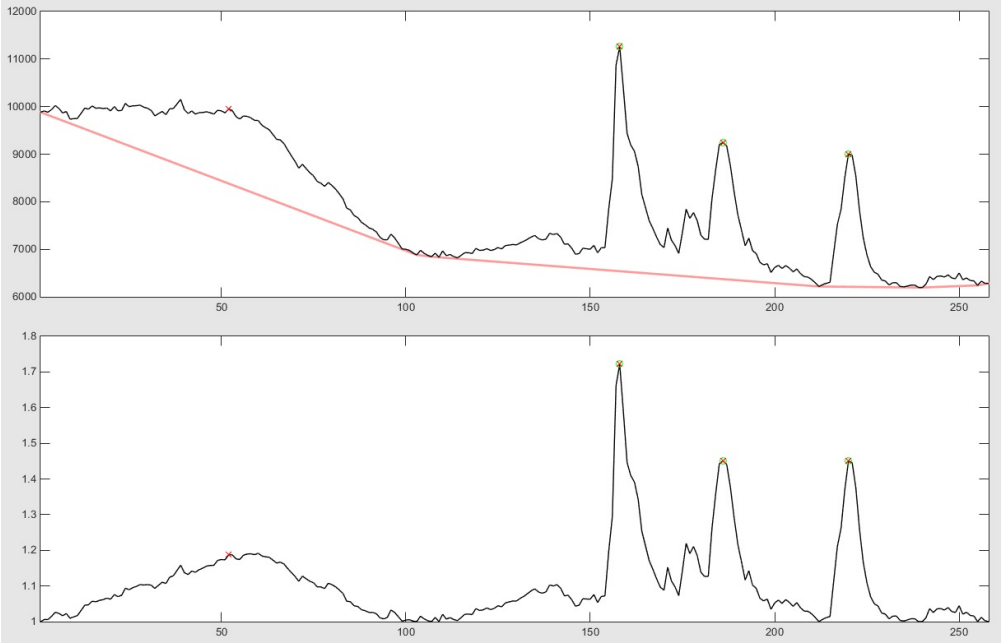


Figure 1.10: These two panels show the actual graphics provided by **PeakCaller**. The top panel shows the example profile, along with the smoothing via the *convex envelope*. The lower graph shows the *de-trended* data series, which is simply the quotient of the original data divided by the smoothed data. Note in this case that the de-trended data "bounces off" of a minimum value of one exactly at the corners where the convex envelope touches the original profile.

Figure 1.8 provides an example calcium profile, along with several smoothings using these *two-sided exponential moving averages*, for increasing values of the *trend smoothness* parameter. A cartoon showing how the values of the original profile are weighted in order to generate the smoothed profile is given in the top panel of Figure 1.9. Notice especially how the weights differ in this case as compared with the two-sided weights given via *finite-difference diffusion* back in Figure 1.5.

Convex Envelope

Suppose that you are holding a tight length of rope beneath the original graph, with one end far off to the left of the graph and the other end far off to the right. You lift the rope, and it eventually touches the graph at its lowest point. With the graph fixed in place, the rope bends around this lowest corner, and you continue to lift both ends of the rope, keeping the rope taut. The rope meets the graph in more places as you lift it, creating additional corners, and eventually you see that the rope connects the left endpoint of the graph to the right endpoint via a number of straight runs – short line segments that join the protruding corners along the bottom of the graph. This collection of segments forms a *convex envelope* that bounds the graph from below.

To define this convex envelope for the original profile $\{x_0^1, x_0^2, \ldots, x_0^T\}$ numerically, we start at the left endpoint x_0^1. The corner we connect this end to is chosen from among all of the points on the graph as the one that *minimizes* the slope of the secant line joining the left endpoint to it. That is, we find the value t_1 satisfying

$$t_1 = \min_{t>1} \ \frac{x_0^t - x_0^1}{t-1} \ .$$

After finding the corner at t_1, as long as $t_1 < T$, we next search for the value (t_2) satisfying

$$t_2 = \min_{t>t_1} \ \frac{x_0^t - x_0^{t_1}}{t-t_1}.$$

We continue identifying additional corners in this way until the end of the interval is reached.

For any time t in the interval, the smoothing at t is given by identifying the point on the line segment that runs beneath the graph and joins the nearest corners to the left and right of t. In particular, if $t_{n-1} < t < t_n$, then the smoothed value at t is given by

$$x_s^t = \left(\frac{t_n - t}{t_n - t_{n-1}}\right) x_0^{t_{n-1}} \ + \ \left(\frac{t - t_{n-1}}{t_n - t_{n-1}}\right) x_0^{t_n} \ .$$

The *smoothness parameter* plays no role in defining the trend in this case, as the smoothing is defined in a parameter-free way by an advanced game

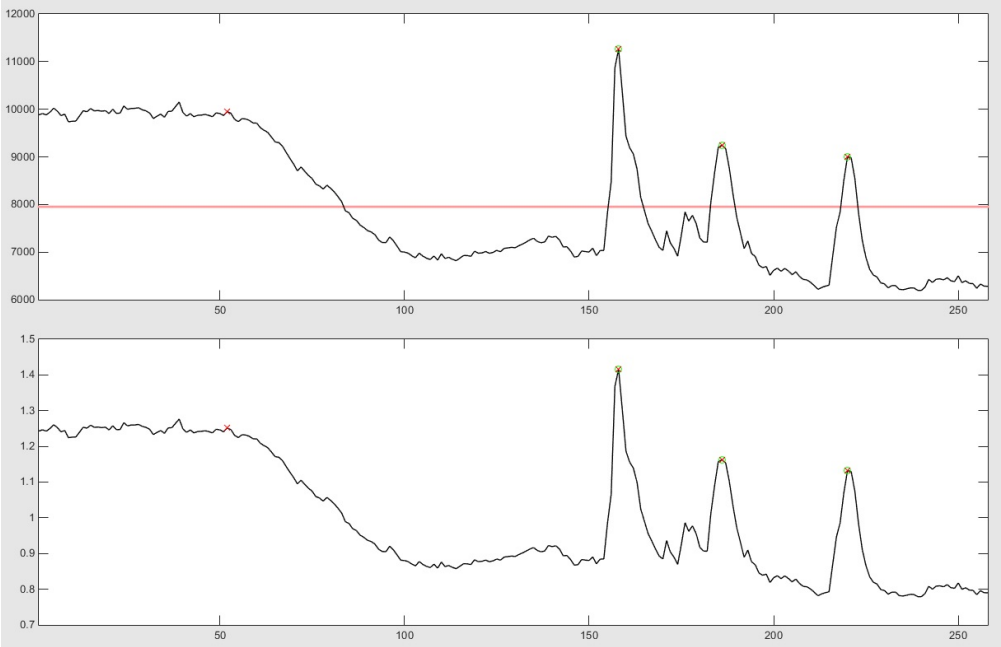


Figure 1.11: These two panels show the actual graphics provided by **PeakCaller**. The top panel shows the example profile, along with the trivial smoothing provided when *no trend* is chosen. The lower graph shows the *not-so-de-trended* data series, which is simply the quotient of the original data divided by the smoothed data. In this case, the de-trended data has exactly the same profile as the original data, but scaled so that the mean is one.

of "connect the dots". The smoothing and the resulting de-trended profile are both displayed in Figure 1.10.

No Trend

If the user is happy with the original data, there is an option to leave it pristine and unmanipulated. When *no trend* is chosen, the *mean* of the original profile is used as a trivial smoothing. The de-trended profile displayed by **PeakCaller** will then have exactly the same shape as the original profile, but scaled so that the mean is one. For the sake of completeness, this not-

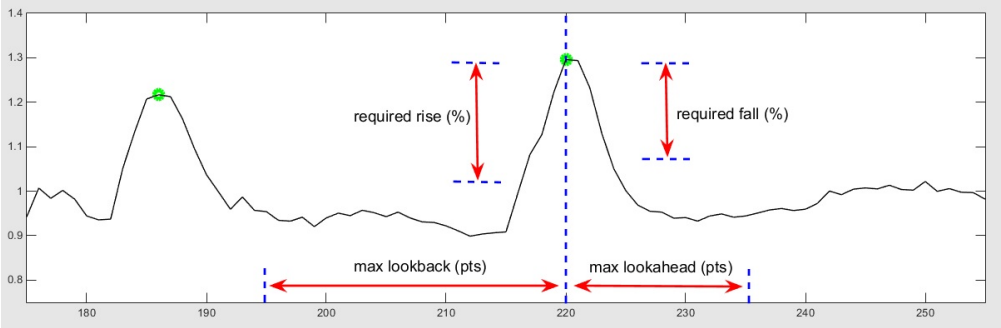


Figure 1.12: This figure details the parameters used by **PeakCaller** to characterize a peak. For a point to be classified as a peak there must be a significant rise over a designated interval before the point, and a significant fall over a designated interval after the point.

so-smooth smoothing, as well as the resulting not-so-de-trended profile, are displayed in Figure 1.11.

1.3 Peak Parameters

PeakCaller characterizes peaks using a clear and straightforward set of criteria.

First, in order for a point to be called a *peak*, the de-trended data must *rise* a significant amount in a well-defined window just *before* the candidate point. The user can define the size of the rise that they deem significant (by setting *required rise (%)* in **PeakCaller**) as well as the maximum length of the window that this rise must occur within (*max lookback (pts)* in **PeakCaller**). The actual lookback window will be shortened appropriately if either (a) the *max lookback* would take us back beyond a previously identified peak or (b) the *max lookback* would go back beyond the beginning of the data.

Second, for a point to be a *peak*, the (de-trended) data must *fall* a significant amount in a well-defined window just *after* the candidate point. The user can define the size of the fall that they deem significant (by setting *required*

fall (%) in **PeakCaller**) – this value does not have to match the previously-defined rise on the other side. The user also defines the maximum length of the window that this fall must occur within (*max lookahead (pts)*). The actual lookahead window will be shortened appropriately if either (a) the *max lookahead* would take us beyond the end of the data or (b) the *max lookahead* would include points that are *higher* than the candidate.

A cartoon showing the four quantities that a user can use to define a peak is given in Figure 1.12. Although many users may choose matching values for rise and fall, as well as lookback and lookahead, **PeakCaller** uncouples these values to allow the user some versatility. In many applications, signals are not perfectly symmetric – with depolarization and repolarization happening on different timescales – and the user now has the freedom to vary these parameters as they deem appropriate. On the more technical side, there is a slight difference in the definitions of the *max lookback* and *max lookahead* that merit a moment's discussion. **PeakCaller** conducts its search for peaks in two sweeps. **PeakCaller** first moves *forward* through the data (from left-to-right), and the forward-looking window is cut off as soon as the data reaches a level that is higher than the current candidate. The backward-looking window can stretch *beyond* higher level, as long as those higher levels were not associated with previous peaks. In its backward sweep, **PeakCaller** moves right-to-left and verifies that the *required fall* following each peak was completed before reaching the following peak.

Graphically, **PeakCaller** denotes a peak by placing a green circle around the point in both the original and the de-trended data.

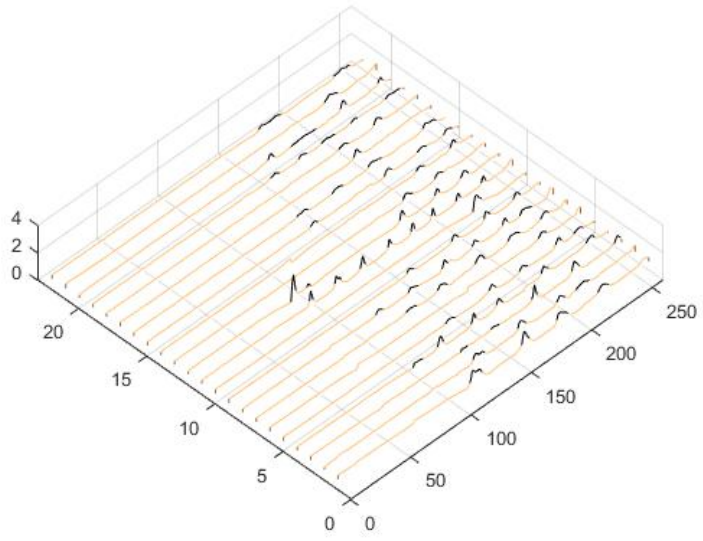


Figure 1.13: *View Traces*. Pressing the button labelled *View Traces* allows the user to see all of the fluorescence traces in a single figure, with the identified peaks highlighted in a different color.

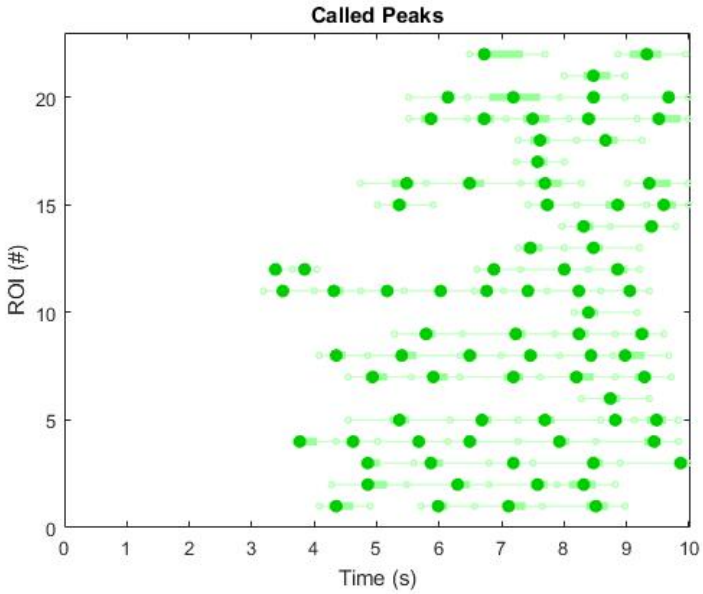


Figure 1.14: *Raster Plot.* Pressing the button labelled *Raster Plot* allows the user to see a raster plot of all of the identified peaks in a single figure.

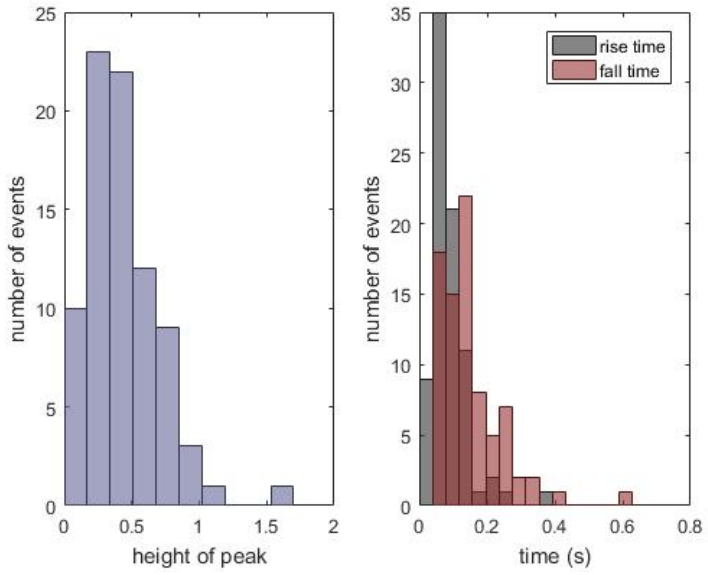


Figure 1.15: *Histograms.* Pressing the button labelled *Histograms* generates a pair of histograms. The chart on the left summarizes the heights of the various peaks found in the file, while the chart on the right summarizes the rise and fall times of those same peaks.

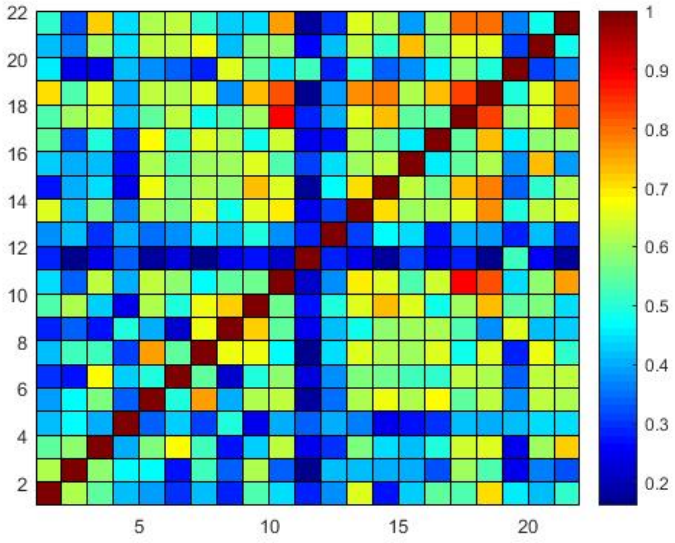


Figure 1.16: *Correlations*. Pressing the button labelled *Correlations* generates a heat map that summarizes the maximum correlation between each pair of fluorescence traces (and their shifts of up to 50 ticks).

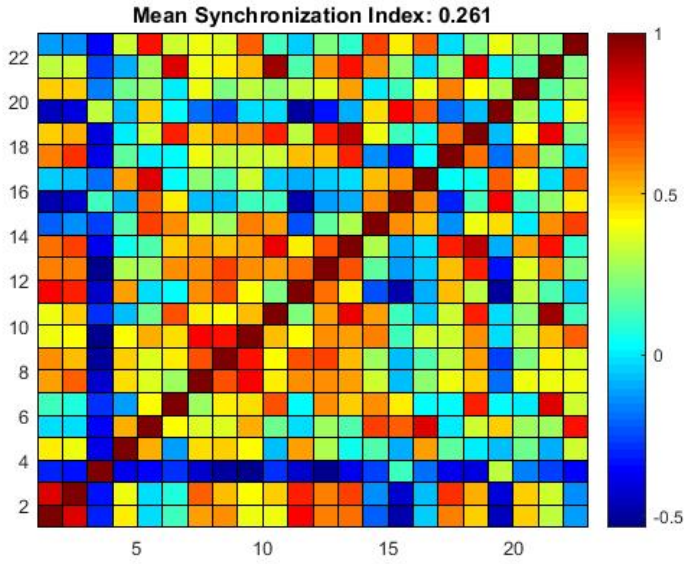


Figure 1.17: *Synchronicity*. Pressing the button labelled *Synchronicity* displays the synchronization matrix based upon the peaks of the different fluorescence traces. It also provides the mean value of the synchronization matrix.