Helpful Tips:

Speed

Different suggestions for how to speed up (significantly) your deconvolution are written next. The solution depends on what is causing the slow down and what things are important to you when you deconvolved. So here is a list of possible reasons for **why the script is slow** and how to speed it up accordingly:

1. **Inverting T*T matrix**: All methods require inverting a matrix that is as big as the length of your traces (T). Depending on your computer this will take very long if your traces are longer than ~600-2000 (typical laptop-typical desktop). If this is the case **you can cut the traces and later concatenate the answers**. In this case if you normalize the deconvolved rate solutions you should normalize them only after concatenating them. I've looked into the results of such an approach (in an earlier version of the manuscript), only the first differences method is noticeably impacted by it because it is the only method that heavily relies on smoothing. You can also cut with overlapping edges of a few points and concatenate back with some running average on the overlapping results. You are very welcome to contact me if you wish me to write a script that does it according to your data structure.

2. **Iterations**: All methods except for the first differences include iterations in their algorithms. If this is your problem you can use the first differences (simply replace convar with firdif in all relevant places along the script if you are following my example script). First differences is not as accurate, but the results differ only in a few percentage, and in return it is 1000-10,000 times faster. There are also differences, due to Iterations, between the different methods. Dynbin iterates separately for every trace, so it is the slowest. Convar iterates all traces in parallel, so it is number-of-traces faster. Lucric in its core uses an algorithm written by matlab so it is better optimized. But the paper shows this is also the poorest performing method, so I advise to avoid it unless you want extremely smoothed results. **Another very fast approach (no iterations, runs in parallel for all traces) is by replacing convar with convar_analytic_unconstraint** everywhere along the script. This will give you unconstrained results (negative rates). You can shift the results to be non-negative. This will give the correct solutions in the vast majority of cases but it is not guaranteed (as we show on the appendices). Or you can just work with negative rates.

3. **Matlab likes real numbers within the range 1-100**. Not sure why, but it runs faster. So if your calcium after DFF is around 0.01 scale, multiple it by a hundred before running the deconvolution. If it is around a different number - multiply accordingly.

Parameters and sensitivity to parameter values

In general the deconvolution algorithms are robust. So if you use slightly different parameter values the results change only slightly. Nothing drastically happens. We show this in figure 5 of the manuscript.

Lambda λ

This parameter sets the weight for the penalty. The penalty is defined by how much the rate is changing across time. The larger lambda is, the larger the penalty weight is, which means that optimization would farther minimize the penalty - the changes in the rate. Hence larger

lambda value means smoother (less changing) deconvolved rate with lower fluctuation level. And hence smaller lambda values give noisier results. **In one sentence, you can think of it as a smoothing parameter.** Setting this parameter is important for correct deconvolved rate since we wish it to overcome noise in the recordings but not lose any information about the neuronal activity (as noisy as the later may be in relation to your objective). The script finds its best value given the noise in the recordings.

Gamma γ, calcium decay

This is the calcium decay between two measurement points. More specifically this is the ratio between the fluorescence at the next time step, compare to the fluorescence at the current time step assuming no spikes occurred between the two time steps (zero rate).

In the example included (also appears in the manuscript), the recordings were done at 40hz and a unit calcium level at one measurement point decays to 0.97 calcium level at the next measurement point (assuming no spikes/ zero rate). Hence gamma = 0.97 in this case. In other words, during 25ms 1 (arbitrary unit) calcium dropped to 0.97. This is a good estimate for GCaMPf mice. In another example in the manuscript, at 10hz measurement rate of GCaMPs I had gamma = 0.95.

How to estimate your calcium decay

It is possible to estimate the calcium decay rate from known single cell calcium response properties (we show it in the manuscript). In the example I bring one of the resources for estimating the decay was the paper Chen et al. "Ultrasensitive fluorescent proteins for imaging neuronal activity". Single cell calcium responses are presented in Chen et al. in figure 1. One can trace the decay shapes (as if one had a pen and a ruler). From such tracing you will discover that the calcium decays to its 0.97 of its original peak in 25ms in GCaMP6f mice and to 0.95 of its original peak in 100ms for GCaMPs mice.

If you are still confused you are welcome to send me the details of your recordings (recoding rate and a link to a study about the calcium response properties) and I do my best to recommend a calcium decay value to use (gamma).

Recording rate

The deconvolution scripts don't care what is your recording rate, only how much the calcium decays from one measurement point to the next. A preferable recording rate (and gamma accordingly) defines time interval between two consecutive measurement points that is longer than, or about, the calcium rise time. If your recordings are very noisy and your recoding rate is very high, you may want to consider averaging over a few time points to get a single new trace with lower recording rate and higher signal to noise ratio (that worked well for me with some of the mice I had).

The calcium baseline

The deconvolution algorithms are blind to a constant shift in the calcium baseline. In the algorithms the mean of the trace is irrelevant (in fact, the first step in each algorithm is to subtract the mean of the trace - the constant calcium baseline).

If the neuronal activity is changing across time this will be detected by the algorithms. If it the calcium trace that has is drifting and hence has a time dependent baseline this will create errors. I assume DF/F was performed on the calcium traces prior to the deconvolution.

Other methods and ROI size

The scripts are blind to whether they are used on a single pixel or an average across a region with multiple pixels as long as the fluorescence trace is a results of many (>>1) neurons' activity. The scripts are also blind to whether de-noising or hemodynamics was performed on the traces. Although we recommend to avoid de-noising prior to deconvolution (see the manuscript appendices).