Self-gating of O4a $h(t)$ for use in continuous-wave searches

1 Motivation and context

Although CW searches suffer their most severe degradation from persistent narrow-band noise artifacts, transient broad-band noise artifacts are not harmless to CW searches. The cumulative effect of many loud transient glitches can raise the noise floor and increase the sensitivity depth necessary to recover CW signals. A strategy to mitigate this effect, known as *self-gating*, flags and removes loud glitches using information solely from the channel to be gated.

Self-gating was used in O3 to improve the noise floor for CW searches. [1] The O3 self-gating procedure used two tests to determine where gates should be a applied: first, a time series of the band-limited RMS (BLRMS) for the range 25-50 Hz was compared to a threshold; second, a time series of the BLRMS for 70-110 Hz was compared to a different threshold. When either threshold value was exceeded, a gate was created. Fixed threshold values were determined for each detector using "an iterative and ad hoc process, based on subjective tradeoffs between livetime loss and spectral distortion."

Subsequently, researchers outside the LVK developed a more sophisticated gating method with a threshold on the amplitude of the absolute whitened strain, where the the threshold value was *not* fixed in advance. [2] Instead, they used an iterative process to establish the amplitude threshold used for gating. Trial thresholds were evaluated by computing a ratio between the gated PSD and a reference PSD, and an amplitude threshold was accepted once this ratio dropped below an established threshold on the ratio.

While the method described in [2] was promising, we did not elect to use it directly in O4. There were a few reasons for this. First, we wanted to control the deadtime (total amount of gated time) more directly to keep it within a nominal value of 1% as requested by members of the LVK CW group. Second, we wanted the option to gate on both BLRMS (as was done in O3) *and* on whitened strain amplitude. Third, because we began development in early O4, we did not know enough about the character of O4 data to evaluate whether the reference PSD method would be effective, so we aimed to avoid dependence on a reference at all. Finally, we considered that Python code built on GWpy would be the best option for maintenance and usability (Python is generally recommended for DetChar tools).

For O4, we have drawn inspiration from both previous methods to develop a new gating procedure described below. The code is written in Python and makes use of many tools available in GWpy. Multiple methods are used to identify gates, testing BLRMS as well as amplitude. A new procedure is introduced for iteratively selecting a threshold within specified deadtime limits, which does not depend on a reference PSD as an end point but rather directly quantifies the PSD improvement relative to ungated case and the tradeoff with livetime loss. [3] All code can be found at `https://git.ligo.org/detchar/tools/dchgate`.

2 Selection and processing of input data

First, an analysis time period of interest is selected (in O4a, gates were computed daily, so this would have a duration of 24 hours). We then select all science segments in the time period of interest using the `{IFO}:DMT-ANALYSIS_READY` flag, and iterate through the science segments, breaking them into "working segments" of maximum length T_{seg} with overlap T_{overlap}.

For each working segment, we compute gates for each of the different threshold methods specified. The O4a production configuration uses three methods: one which gates on amplitude, and two which gate on BLRMS in different frequency bands. The gates from the different methods are coalesced (the union is taken) so that a time period will be gated if *any* of the gate methods detected a glitch within it.

Next, the resulting gates from all working segments are coalesced. This ensures that gates are properly combined in the time regions of overlap between working segments.

2.1 Start and end padding times [1]

The analyzed time period is extended by a short duration $T_{\text{pad}}$to ensure that there is no discontinuity at the boundaries between analyzed times, nor between working segments. At the end of the procedure, we also crop the list of gates and the data itself back to the original extent of the analyzed time period so that no extra data is retained.

2.2 Data preparation

Before gating, the data in the working segment is whitened using the `whiten` method in `gwpy` with default parameters.[2] The whitened amplitude will be used directly for the amplitude gating stages. It is also the input for a BLRMS calculation in each of the specified frequency bands.[3] The BLRMS calculation also uses `gwpy` methods, specifically `bandpass` and `rms`. The BLRMS calculation is done for a specified frequency range $f_{\text{BLRMS}}^{\{\text{min,max}\}}$ with a specified stride in seconds t_{stride}.

2.3 Parameter reference for selection and processing of input data

Parameter	Argument name	Description	Value used for O4a	Notes
$T_{\text{pad}}^{\{\text{start,end}\}}$	`pad-time-{start,end}`	Padding time at the start and end of the analyzed time	5 s	None
T_{seg}	`segment-step`	Length of working segments in which to compute gates	1024 s	None
T_{overlap}	`segment-overlap`	Overlap of working segments	64 s	None
$f_{\text{BLRMS}}^{\{\text{min,max}\}}$	`blrms-f{low,high}`	Frequency range to use for BLRMS calculation	25-50 Hz and 70-110 Hz	Threshold will be established and gating performed for *each* range specified.
t_{stride}	`blrms-step`	Stride to use for BLRMS calculation. See `rms` method in `gwpy`.	0.0625 s	Allowed to differ for each frequency range, but O4a configuration uses the same for each

[1] This section refers to the padding of the over all analyzed time period. Note that there is a different padding time, t_{pad} referenced later which refers to the padding time for individual gates. These are unrelated quantities.

[2] Default whitening: FFT length is selected based on sample rate. A "constant detrend" is applied to the data before the FFT (only the mean of data is subtracted). Hann windowing with 50% overlap is applied to the data before the FFT. The "averaging" method is a median. The time-domain FIR whitening filter used has a duration of 2 seconds. Additional parameter information can be found in the GWpy documentation.

[3] Alternatively, whitening can be turned off for the BLRMS calculation.

3 Gates and deadtime

This section will discuss how gates are applied for a *given* threshold R; threshold selection will be discussed in detail in section 4. Note also that the gating procedure is identical whether the threshold in question is on BLRMS or on whitened amplitude.

The actual identification of gates is done using the `gwpy` method `find_gates`. This yields a list of time segments to zero. To avoid spectral artifacts from discontinuities, a smoother window is needed. We use a Planck window, which tapers down to zero over a duration $t_{\rm pad}$before the gate, then back up to one over a duration $t_{\rm pad}$ after the gate.

3.1 Details of overlapping window functions

When the gates returned by `find_gates` overlap, this can be handled easily by coalescing them. When the gates returned by `find_gates` are separated by at least $2t_{\rm pad}$, there is no need to handle overlapping window functions. However, the intermediate case, where the gate separation is less than $2t_{\rm pad}$, must be handled with care. By multiplying the two overlapping window functions, rather than stitching them together, we can achieve a smooth curve in the overlap region. This situation is illustrated in figure 1.

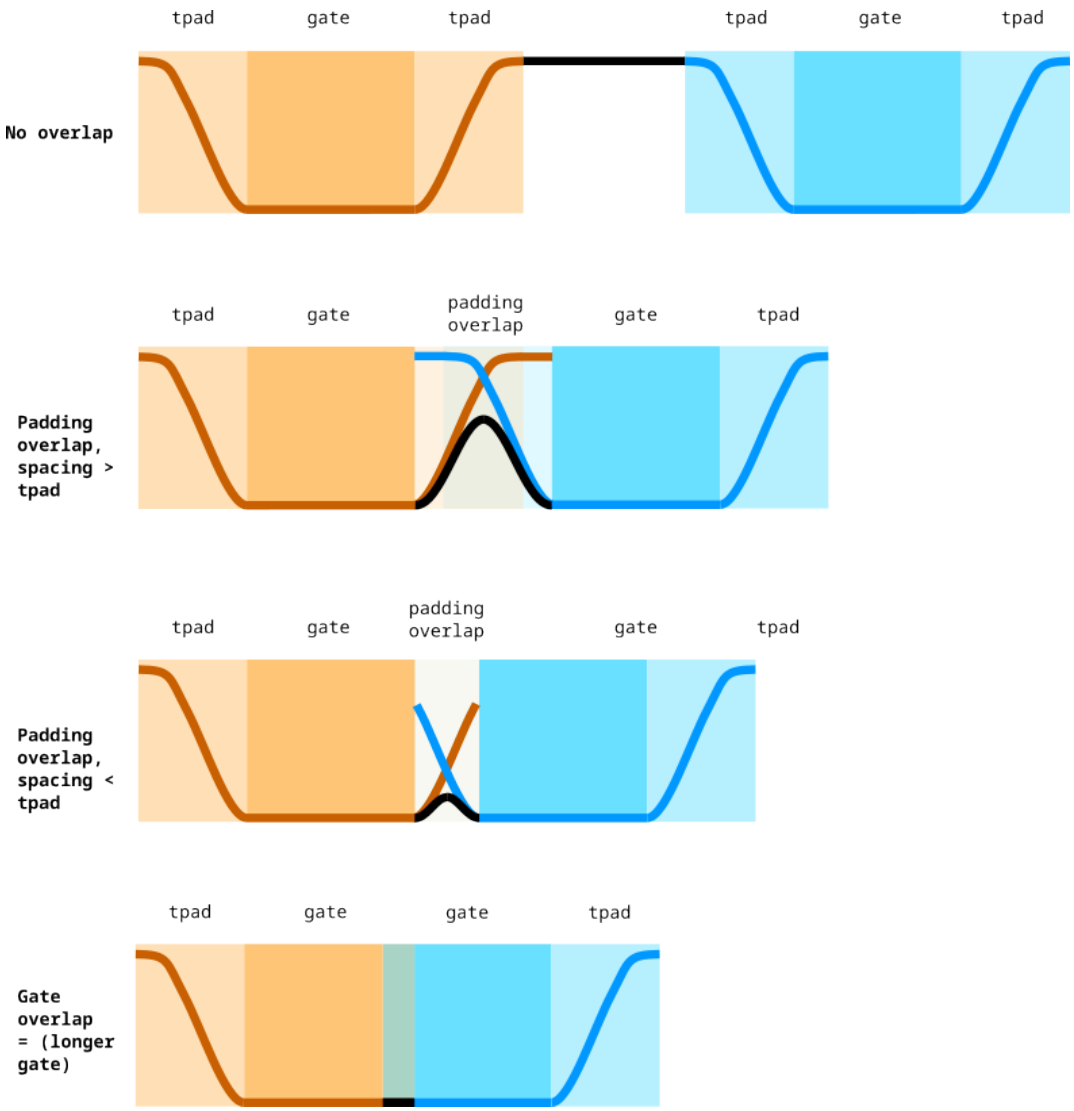

Figure 1: A schematic of gate overlap scenarios. The first scenario requires no special calculations; the last scenario is equivalent to coalescing the zeroed segments for two gates and happens automatically. The second and third scenarios involve an overlapping region of the window functions region. In the overlap region, the brown and blue curves represent the truncated sections of the two window functions, and the smoother black section represents their product. (This is strictly a schematic; the curves are sketched and not calculated.)

3.2 Parameter reference for application of gates

Parameter	Argument name	Description	Value used for O4a	Notes
t_{pad}	`gate_tpad`	Padding time for each gate	0.5 s	None
t_{zero}	`gate-tzero`	`tzero` parameter for `gwpy` method `find_gates`	0.0625	$2t_{\mathrm{zero}}$ is the minimum length for gates returned

3.3 Computation of deadtime

Because tapering is applied outside of the identified gates, the total time in which data is affected by gating exceeds the sum of the gate durations. We must define what counts as "deadtime" (data removed by gating) for the purpose of evaluating various trial thresholds. The current definition is that *all* of the impacted time counts as deadtime, including the entire duration of the taper. In the future, we plan to make the fraction of the taper time counting as deadtime an adjustable parameter.

4 Threshold selection

4.1 Iteration rounds and trials

The threshold is selected iteratively for each of the different gating types specified in the configuration file (whether BLRMS or amplitude gating). Threshold selection begins with a maximum and minimum threshold to test for the given type of gating, $R^{\text{min}}_{\text{initial}}$ and $R^{\text{max}}_{\text{initial}}$. There is a nested loop in the iteration process: first, we iterate over n_{rounds} "rounds", progressively narrowing the range of tested thresholds so that each round tests a threshold range $R^{\text{min}}_{\text{round}}$ to $R^{\text{max}}_{\text{round}}$, where $R^{\text{min}}_{\text{initial}} \leq R^{\text{min}}_{\text{round}} < R^{\text{max}}_{\text{round}} \leq R^{\text{max}}_{\text{initial}}$. Second, within each round, n_{trials} trial thresholds $R_{\text{trial}}$are calculated, spanning between $R^{\text{min}}_{\text{round}}$ and $R^{\text{max}}_{\text{round}}$and equally spaced in $\log R_{\text{trial}}$. The best trial threshold is calculated, and the range of tested thresholds for each subsequent round is determined by the trial threshold value immediately above and below the winning threshold for the current round. An example is illustrated in figure 2.

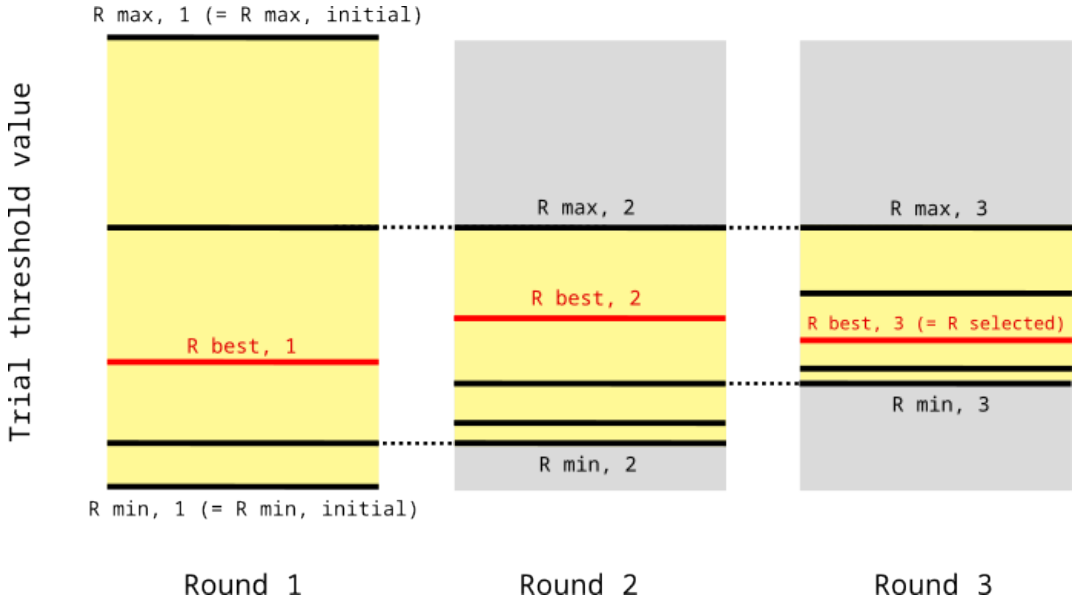

Figure 2: Example of iterating through rounds with $n_{\text{rounds}} = 3$ and $n_{\text{trials}} = 5$. The yellow region is a visual highlight showing how the range of tested thresholds narrows with each round. Note that trial thresholds are spaced logarithmically, which is not represented to scale on this diagram.

4.2 Benefit calculation

In order to determine the "best" threshold for each round, or over all, we require a metric for the benefit of gating with a given threshold. Gating ought to improve (lower) the PSD across a broad frequency range without incurring too much deadtime.

We can quantify the PSD improvement in the frequency band of interest as follows. For some threshold R, the PSD improvement I_R is considered to be:

$$I_R = \text{median}\left(\frac{P_{\text{ungated}}(f) - P_{\text{gated,R}}(f)}{P_{\text{ungated}}(f)}\right),\ f^{\text{min}}_{\text{eval}} \leq f \leq f^{\text{max}}_{\text{eval}}$$

where $P(f)$ is the PSD and $f^{\{\text{min,max}\}}_{\text{eval}}$indicates the frequency range of interest, and the median is computed across frequency bins f.

Next, the deadtime for the given threshold R is calculated, here denoted d_R. The improvement I_R and the deadtime d_R are combined to create a measure of over all *benefit* B_R for the given threshold,

$$B_R = \begin{cases} \frac{(I_R)^w}{\max(d_R, d_{\text{ignore}})} & d_R \leq d_{\text{limit}} \\ 0 & d_R > d_{\text{limit}} \end{cases}$$

where d_{ignore} is a trivially low amount of deadtime under which we choose to ignore the effect of deadtime entirely, and where w is an exponent used to set the weight of the PSD improvement relative to deadtime increase. The maximum allowable deadtime is d_{limit}; above this point, the benefit is zero regardless of how much the PSD has decreased.

4.2.1 Why use d_{ignore}?

Suppose that threshold R_1 generates a very low deadtime percentage, $d_{R_1} = 0.01\%$ and another threshold R_2 generates a similarly low $d_{R_2} = 0.02\%$. Both of these values are negligible for search purposes, and if $I_{R_2} > I_{R_1}$ we should likely use R_2 despite the fact that its deadtime is technically twice as high. The parameter d_{ignore} sets the point at which we start to consider changes in deadtime as non-negligible.

4.3 Parameter reference for threshold setting

Parameter	Argument name	Description	Value used for O4a	Notes
d_{limit}	`deadtime-max-allow-pct`	Deadtime limit: do not accept thresholds that exceed this limit	1.0%	Expressed as a percentage of total time analyzed.
d_{ignore}	`deadline-ignore-pct`	Deadtime percentage to ignore when calculating benefit	0.05%	Expressed as a percentage of total time analyzed.
$f_{\text{eval}}^{\{\text{min,max}\}}$	`psd-test-{fmin,fmax}`	Frequency range to use for PSD improvement calculation	0-300 Hz	None
w	`psd-weight-exp`	Exponent used in benefit calculation to set the weight (importance) of the PSD improvement.	2	When set higher, the PSD improvement matters more relative to deadtime
$R_{\text{BLRMS}}^{\{\text{min,max}\}}$	`blrms-threshold-{min,max}`	Range of BLRMS thresholds to test	0.1-100	If only one value is given, use that as a fixed threshold and do not iterate
$R_{\text{amp}}^{\{\text{min,max}\}}$	`amp-threshold-{min,max}`	Range of amplitude thresholds to test	0.1-1000	If only one value is given, use that as a fixed threshold and do not iterate. Only used if `amplitude-cut` is `True`.
n_{rounds}	`num-rounds`	Number of rounds in which to generate and test progressively narrower ranges of threshold values	3	Outer loop
n_{trials}	`trials-per-rd`	Number of trial thresholds to test per round	5	Inner loop

5 Output

The output of this procedure includes both gated frames, and a list of gate times. Summary pages are also generated with statistics on the gates identified. Note that the writing of frames is done iteratively: the function that writes each frame calls itself again to write the next frame, until the remaining time is less than the maximum frame length.

6 Tests and validation

This section to be filled out.

References

[1] Keith Riles and John Zweizig. Information on self-gating of $h(t)$ used in O3 continuous-wave and stochastic searches. `https://dcc.ligo.org/LIGO-T2000384`.

[2] Benjamin Steltner, Maria Alessandra Papa, and Heinz-Bernd Eggenstein. Identification and removal of non-gaussian noise transients for gravitational-wave searches. *Physical Review D*, 105(2), January 2022.

[3] GWpy: a python package for gravitational-wave astrophysics.
`https://gwpy.github.io`
`https://doi.org/10.5281/zenodo.597016`.