Low-latency Noise Mitigation Techniques in Gravitational-wave Detector Data Using Auxiliary Sensor Information

Patrick Godwin
Department of Physics, Pennsylvania State University
Open Public Alert Timeline

Time since gravitational-wave signal

- Original Detection
- Set Preferred Event
- Automated Vetting
- Classification
- Rapid Sky Localization

Preliminary Alert Sent

Parameter Estimation
- Human Vetting
- Classification

Initial Alert or Retraction Sent

Parameter Estimation

Classification

Update Alert Sent

10 second 1 minute 1 hour 1 day 1 week

Open Public Alert Timeline
Simplified Low-Latency Dataflow: O3

Introduction

Detector

Noise Extraction

Statistical DQ

Calibration

GW Searches

Detections

0(1 s)

0(5 s)

0(20 s)
Detector Noise

Detector → Noise Extraction → Statistical DQ → GW Searches → Detections

0(1 s) → 0(5 s) → 0(20 s)
Noise Types

Strain noise $[1/\sqrt{\text{Hz}}]$

- $10^{-19}$
- $10^{-20}$
- $10^{-21}$
- $10^{-22}$
- $10^{-23}$
- $10^{-24}$

Frequency [Hz]

- 10
- 100
- $10^3$

Noise types diagram showing frequency and time distribution.

Noise Sources

Detector Noise

arXiv:1901.05093v2
Auxiliary Sensors

$h(t)$

wind

microphone

CQG 28, 13 (2012)
Auxiliary Sensors: A Big Data Challenge

- ~250k auxiliary channels
  - ~5k “fast” channels (≥ 256 Hz timeseries)
  - Rest are “slow” channels (16 Hz timeseries)
- Data rate: ~50 MB/s
- Data storage: ~ 1.5 PB/yr
Noise Mitigation Techniques

In order of preference:

1. Noise Source Removal
2. Noise Subtraction
3. Noise Identification
4. Search-Specific Mitigation
Noise Mitigation Techniques

In order of preference:

1. Noise Source Removal
2. Noise Subtraction
3. Noise Identification
4. Search-Specific Mitigation
SNAX Toolkit

Detector → Noise Extraction → Statistical DQ → GW Searches → Detections

- 0(1 s)
- 0(5 s)
- 0(20 s)
SNAX Toolkit: Overview

- Stream-based Noise Acquisition and eXtraction
- Extract non-stationary noise features in low-latency
- Sine-Gaussian basis
- **Output**: Multivariate timeseries data containing non-stationary noise features:

\[
\vec{a}(t) = (\vec{a}_1(t), \ldots, \vec{a}_n(t))
\]

\[
\vec{a}_i(t) = (\rho, \phi, t_{\text{match}}, f, Q)
\]
SNAX Toolkit: Workflow

Signal conditioning:
- Auxiliary channel timeseries
  - High-pass data
  - Whiten data
  - Measure PSD

Split channel into multiple frequency bands:
- No downsampling (2048 Hz)
- Downsampling (1024 Hz)
- Downsampling (128 Hz)

Matched filter waveforms
- Recorded match with max(SNR)
- Generate Buffers
About Latency

![Graph showing frequency vs. time with normalized energy scale.](image-url)
SNAX Toolkit: Latency Budget

Signal conditioning

Auxiliary channel timeseries → High-pass data → Whiten data → Measure PSD

Split channel into multiple frequency bands

No downsampling (2048 Hz)
Downsample (1024 Hz)
Downsample (128 Hz)

Matched filter waveforms → Record match with max(SNR) → Generate Buffers
SNAX Toolkit: Latency Budget

Signal conditioning

Auxiliary channel timeseries → High-pass data → Whiten data

Measure PSD

Split channel into multiple frequency bands

No downsampling (2048 Hz)

Downsample (1024 Hz)

Downsample (128 Hz)

Matched filter waveforms

Record match with max(SNR)

Generate Buffers

1 s
SNAX Toolkit: Latency Budget

Signal conditioning

1 s + 1 s

Auxiliary channel timeseries → High-pass data → Whiten data → Measure PSD → Matched filter waveforms → Record match with max(SNR) → Generate Buffers

No downsampling (2048 Hz)
Downsample (1024 Hz)
Downsample (128 Hz)

Split channel into multiple frequency bands
SNAX Toolkit: Latency Budget

Signal conditioning

Auxiliary channel timeseries → High-pass data → Whiten data → Measure PSD → Split channel into multiple frequency bands

- No downsampling (2048 Hz)
- Downsampling (1024 Hz)
- Downsampling (128 Hz)

Matched filter waveforms → Record match with max(SNR) → Generate Buffers

1 s + 1 s + 1 s
SNAX Toolkit: Latency Budget

Signal conditioning

- Auxiliary channel timeseries
- High-pass data
- Whiten data
- Measure PSD
- No downsampling (2048 Hz)
- Downsampling (1024 Hz)
- Downsampling (128 Hz)

Split channel into multiple frequency bands

- Matched filter waveforms
- Record match with max(SNR)
- Generate Buffers

1 s + 1 s + 1 s + 1 s + 1 s
SNAX Toolkit: Latency Budget

Signal conditioning

Auxiliary channel timeseries → High-pass data → Whiten data → Measure PSD

Split channel into multiple frequency bands

No downsampling (2048 Hz)
Downsample (1024 Hz)
Downsample (128 Hz)

Matched filter waveforms

Record match with max(SNR)

Generate Buffers

Total: 4 s
iDQ

Detector -> Noise Extraction -> Statistical DQ

Calibration -> GW Searches -> Detections

0(1 s) -> 0(5 s) -> 0(20 s)
iDQ: Overview

- Search for correlations between auxiliary channels and $h(t)$
- **Output**: Probability that there is a glitch in $h(t)$ as a function of time:

$$p_G(t) = p(G|\tilde{a}(t)) = \frac{p(\tilde{a}|G)p(G)}{p(\tilde{a}|G)p(G) + p(\tilde{a}|C)p(C)}$$
iDQ: Workflow

**iDQ**

- **train** → **models** (dimensional reduction and feature importance)
- **evaluate** → **quivers** (cross-validation and generalization error)
- **calibrate** → **maps** (probabilistic statements and uncertainty)
- **timeseries** → **predictions** (input for searches and candidate follow-up)

**IFO**

- Channel → Features
- Channel → Features
- Channel → Features
- Channel → Features

**Feature Extractor**

**segments**
iDQ: Workflow

Features
\[ \vec{a} \in \mathcal{R}^N \]

Models
\[ \mathcal{M}_\alpha : \vec{a} \in \mathcal{R}^N \rightarrow r_\alpha \in [0, 1] \]

Maps
\[
\begin{align*}
p(r_\alpha | G) \\
p(r_\alpha | C')
\end{align*}
\]

Predictions
\[
\begin{align*}
\Lambda_C^G(r_\alpha) &= \frac{p(r_\alpha | G)}{p(r_\alpha | C')}
\end{align*}
\]
\[
\begin{align*}
p_G &= \frac{p(r_\alpha | G)p(G)}{p(r_\alpha | G)p(G) + p(r_\alpha | C)p(C')}
\end{align*}
\]
iDQ: Features

- Automatic retraining to deal with detector non-stationarity
- Provenance tracking
- Input features are relatively backend agnostic (SNAX, Omicron, etc)
- Supports variety of supervised classifiers
  - Statistical veto algorithms (OVL)
  - Classical ML classifiers (scikit-learn)
  - Gradient boosting (XGBoost)
  - Deep learning (Keras, PyTorch)
Ordered Veto List

1. Apply veto configuration with a given threshold/window
2. Calculate veto efficiency for said configuration
3. Repeat for all threshold/window combinations for all auxiliary channels
4. Trim down set of veto configurations hierarchically
iDQ: GW170817

arXiv:2005.12761v1
iDQ: RF Whistle

arXiv:2005.12761v1
Wishlist For Online DQ

- Automated noise identification + mitigation (<5 second latency)
  - Medium-duration noise mitigation
  - Strain-only noise transient source mitigation
  - Noise subtraction for short duration transients
- Incorporation of non-binary DQ information into online searches
- Lock loss prediction