SoftwareX 14 (2021) 100680

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

GstLAL: A software framework for gravitational wave discovery

Kipp Cannon [a], Sarah Caudill [b,p], Chiwai Chan [a], Bryce Cousins [c,d], Jolien D.E. Creighton [e], Becca Ewing [c,f], Heather Fong [a,g], Patrick Godwin [c,f], Chad Hanna [c,f,h,d], Shaun Hooper [i], Rachael Huxford [c,f], Ryan Magee [m,c,f,*], Duncan Meacher [e], Cody Messick [q], Soichiro Morisaki [j], Debnandini Mukherjee [c,f], Hiroaki Ohta [a], Alexander Pace [c,f], Stephen Privitera [k], Iris de Ruiter [l,b], Surabhi Sachdev [c,f,m], Leo Singer [n], Divya Singh [c,f], Ron Tapia [c,d], Leo Tsukada [a,g], Daichi Tsuna [a,g], Takuya Tsutsui [a], Koh Ueno [a], Aaron Viets [e], Leslie Wade [o], Madeline Wade [o]

[a] *RESCEU, The University of Tokyo, Tokyo, 113-0033, Japan*
[b] *Nikhef, Science Park, 1098 XG Amsterdam, Netherlands*
[c] *Department of Physics, The Pennsylvania State University, University Park, PA 16802, USA*
[d] *Institute for Computational and Data Sciences, The Pennsylvania State University, University Park, PA 16802, USA*
[e] *Leonard E. Parker Center for Gravitation, Cosmology, and Astrophysics, University of Wisconsin-Milwaukee, Milwaukee, WI 53201, USA*
[f] *Institute for Gravitation and the Cosmos, The Pennsylvania State University, University Park, PA 16802, USA*
[g] *Graduate School of Science, The University of Tokyo, Tokyo 113-0033, Japan*
[h] *Department of Astronomy and Astrophysics, The Pennsylvania State University, University Park, PA 16802, USA*
[i] *University of Western Australia, Crawley, Western Australia 6009, Australia*
[j] *Institute for Cosmic Ray Research, The University of Tokyo, 5-1-5 Kashiwanoha, Kashiwa, Chiba 277-8582, Japan*
[k] *Albert-Einstein-Institut, Max-Planck-Institut für Gravitationsphysik, D-14476 Potsdam-Golm, Germany*
[l] *Astronomical Institute, Anton Pannekoek, University of Amsterdam, Science Park 904, 1098 XH Amsterdam, Netherlands*
[m] *LIGO Laboratory, California Institute of Technology, MS 100-36, Pasadena, CA 91125, USA*
[n] *Astroparticle Physics Laboratory, NASA Goddard Space Flight Center, Mail Code 661, Greenbelt, MD 20771, USA*
[o] *Department of Physics, Hayes Hall, Kenyon College, Gambier, OH 43022, USA*
[p] *Institute for Gravitational and Subatomic Physics (GRASP), Department of Physics, Utrecht University, Princetonplein, 13584 Utrecht, Netherlands*
[q] *The Center for Gravitational Physics, University of Texas, Austin, TX 78712, USA*

ARTICLE INFO

Article history:
Received 1 June 2020
Received in revised form 4 December 2020
Accepted 9 February 2021

Keywords:
Gravitational waves
Neutron stars
Black holes
Multi-messenger astrophysics
Data analysis

ABSTRACT

The GstLAL library, derived from Gstreamer and the LIGO Algorithm Library, supports a stream-based approach to gravitational-wave data processing. Although GstLAL was primarily designed to search for gravitational-wave signatures of merging black holes and neutron stars, it has also contributed to other gravitational-wave searches, data calibration, and detector-characterization efforts. GstLAL has played an integral role in all of the LIGO-Virgo collaboration detections, and its low-latency configuration has enabled rapid electromagnetic follow-up for dozens of compact binary candidates.

Code metadata

Current code version	1.7.3
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2020_238
Code ocean compute capsule	N/A
Legal code license	gnu general public license
Code versioning system used	git
Software code languages, tools, and services used	python, c, sqlite
Compilation requirements, operating environments & dependencies	scientific linux 7, python 2.7
If available link to developer documentation/manual	https://lscsoft.docs.ligo.org/gstlal/
Support email for questions	gstlal-discuss@ligo.org

* Corresponding author at: LIGO Laboratory, California Institute of Technology, MS 100-36, Pasadena, CA 91125, USA.
E-mail address: rmmagee@caltech.edu (Ryan Magee).

https://doi.org/10.1016/j.softx.2021.100680

1. Motivation and significance

Gravitational waves were originally predicted by Einstein in 1916 [1] as a consequence of general relativity, which describes gravity as the warping of space and time caused by mass and energy [2]. Two extremely massive objects orbiting one another e.g., black holes or neutron stars, warp space dynamically and send ripples across the universe that can be observed here on Earth. As they pass by, gravitational waves stretch and squeeze the space around Earth by less than the width of an atom compared to the Earth's diameter. Due to their tiny effect on scientific instruments, gravitational waves were not observed until 100 years after their initial prediction. Technological advances in laser interferometry led to the discovery of gravitational waves from a merging binary black hole in 2015 [3]. This watershed moment was made possible by the Advanced Laser Interferometer Gravitational-wave Observatory (LIGO) [4] and the scientists of the LIGO and Virgo Collaborations. The effect of passing gravitational waves and the goals of the GstLAL framework are described in Fig. 1.

Advanced LIGO, Advanced Virgo, and KAGRA [4–7] are the currently operating worldwide network of kilometer-scale laser interferometric gravitational wave observatories which have measured gravitational wave signals. These detectors provide a new way to observe our Universe and enable a vast amount of new science. Advanced LIGO-Virgo observations have already deepened our understanding of the populations of compact objects such as neutron stars and black holes [8,9], and they have also offered new tests of fundamental physics [10–13]. The strong gravity regime probed by compact binary mergers is a laboratory for novel tests of general relativity, and a joint observation of gravitational waves along with electromagnetic waves [14] has taught us more about how matter behaves in the most extreme conditions [15,16].

The science made possible by LIGO and Virgo is reliant on measuring miniscule changes in the relative arm lengths of the interferometers known as strain. The perturbations caused by incident gravitational waves manifest themselves as variations in the intensity of laser light output. *Detector calibration* aims to accurately map the intensity of the output to differential changes in the arm length through real-time signal-processing [17,18]. The calibrated strain data contain the encoded properties of the astrophysical systems that produce gravitational waves. The analysis of these data is complicated by the presence of a vast array of transient noise sources. *Detector characterization* aims to quantify departures from stationary noise to identify times where instrumental issues are so severe that the data should not be analyzed or there may be a coupling between environmental sensors (such as seismometers) and the gravitational wave strain data [19,20]. Once data are calibrated and assessed for quality, they are analyzed by a host of detection algorithms to identify potential gravitational wave signals. In many cases the signals are invisible to the naked-eye in raw data and discovering them requires sophisticated techniques that may involve checking millions of models against each segment of data. All three of these activities require substantial cyberinfrastructure. The GstLAL software framework [21] was initially designed to support low-latency compact binary searches to facilitate multi-messenger astronomy, but since its conception it has grown to be a key component of the software used to produce accurately calibrated strain data [17], and recently it has contributed to detector characterization efforts [22,23]. The GstLAL framework is now contributing key cyberinfrastructure to all three of these key aspects of gravitational wave data analysis. This paper will describe how the GstLAL software is used in gravitational-wave searches [24,25], provide examples, and describe the history and impact of GstLAL on gravitational wave discovery.

2. Software description

Gravitational-wave strain data quantify how the distance between two points will change as a gravitational wave passes. The current gravitational wave observatories are sensitive to changes in strain and measure the stretching and squeezing of space as a function of time. The Advanced LIGO [4] and Advanced Virgo [5] gravitational wave detectors are most sensitive to strain frequencies between 10 Hz–8 kHz, which is remarkably close to the frequency range of the human ear [26]. For this reason, there is a close connection between the analysis of gravitational-wave data and the analysis of audio data. Indeed, techniques such as low pass filtering, high pass filtering, channel mixing, and gating apply equally well to both audio processing and gravitational-wave data processing, which provides the motivation for basing GstLAL on Gstreamer [27].

Gstreamer [27] is an open-source, cross-platform multimedia processing framework designed to execute audio and video processing graphs organized into three basic elements: sources, filters and sinks, which are provided by dynamically loaded plugins. A valid Gstreamer graph, called a pipeline, connects elements together ensuring that the capabilities of each element are satisfied. The data are passed along in *buffers* that store both the memory location of the raw data as well as rich metadata. Pipelines can be used to construct complex workflows and scale to thousands of elements. GstLAL combines standard Gstreamer signal processing elements with custom elements to analyze LIGO strain data. An example Gstreamer graph is shown in Fig. 2.

The GstLAL software began development in 2008 through the exploration of novel techniques for filtering gravitational wave data [28]. It derives its name from "Gstreamer wrappings of the LIGO Algorithm Library[1] (LAL)". GstLAL began to take on its modern form by 2009 and has been actively developed as open source software since. GstLAL currently resides in the LIGO Scientific Collaboration hosted GitLab instance at https://git.ligo.org/lscsoft/gstlal [21].

GstLAL is primarily a mix of Python and C with contributions from 75 authors distributed across North America, Europe, Asia and Australia. The master branch currently has over 13,000 commits and 250,000 lines of code. GstLAL is released under the GPLv2 license with 44 distinct releases since 2011 [29]. In 2012, code solely used for gravitational wave searches for compact binaries was split off into its own package: GstLAL-Inspiral. GstLAL-Inspiral has had 45 distinct releases since then. In 2014, code used for gravitational wave burst detection was split into its own package, GstLAL-Burst, with nine distinct releases. And finally, in 2014 code used primarily for LIGO strain data calibration was split into its own package, GstLAL-Calibration, with 60 releases. In addition to tar-ball releases, RedHat and Debian compatible packages are produced for the LIGO Data Grid reference platforms [30].

At present, Docker containers with the full GstLAL/LALSuite software stack are built and distributed using the LIGO Container Registry [31]. The containers are built on top of the CentOS-based Scientific Linux 7, which currently serves as the reference operating system on the LIGO Data Grid. Binary executables are linked against Intel's high-performance Math Kernel Library (MKL), and compiled to leverage Advanced Vector Extensions. Optimized versions of the GstLAL software stack tuned at the compiler-level to best leverage the native features of the local computing environment are often custom-built by users. GstLAL is also available through CondaForge [32].

[1] https://git.ligo.org/lscsoft/lalsuite.

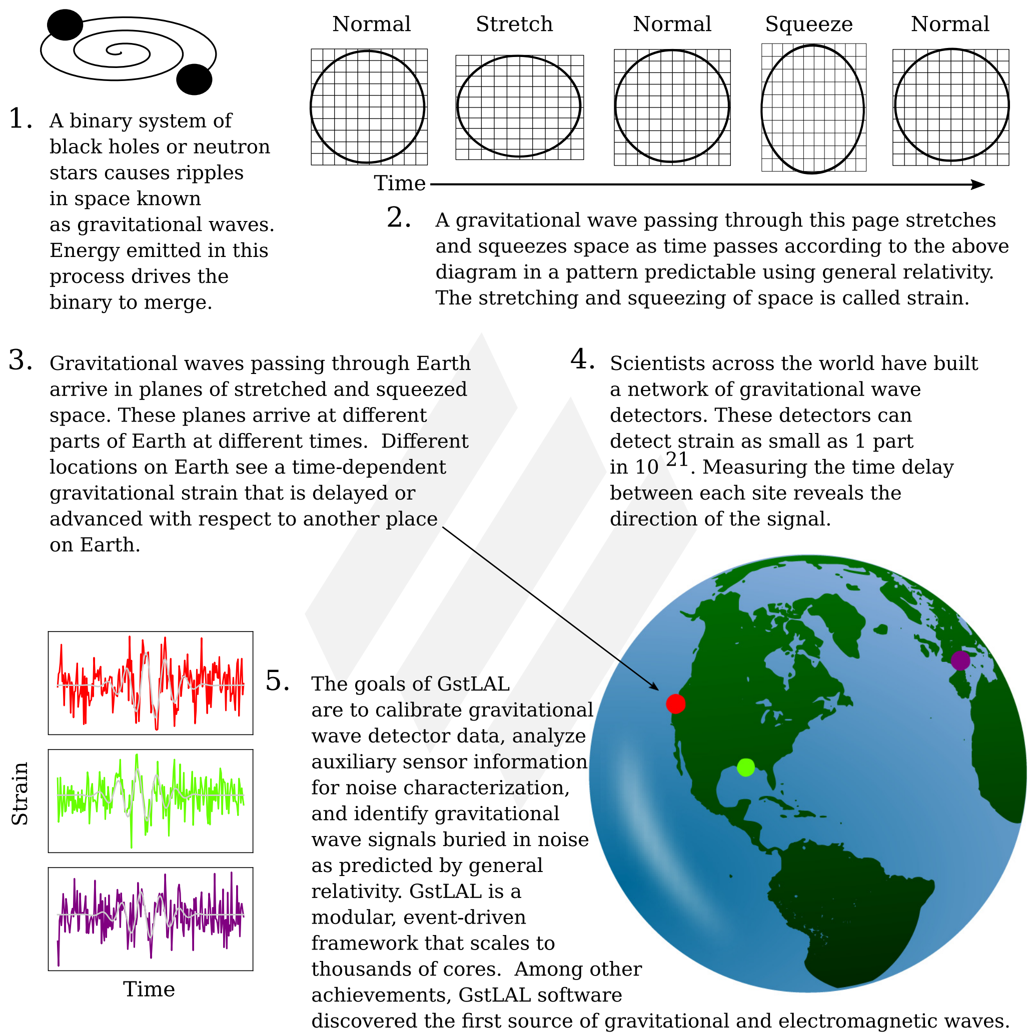

Fig. 1. Gravitational wave infographic. Gravitational wave data are time series, audio frequency data that are noise dominated. GstLAL identifies signals consistent with the predictions of general relativity as measured by multiple gravitational wave detectors and assesses the probability that these signals come from merging neutron stars and/or black holes in near real-time.

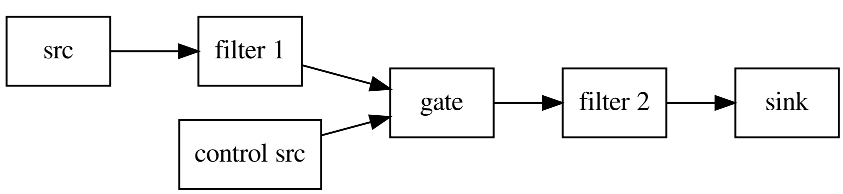

Fig. 2. A basic Gstreamer graph. Data starts at a source, "src", e.g., a file on disk or a network socket, and then is passed through a filter element, "filter 1", which transforms the data, e.g., by performing a low pass filter. A second data stream starts from "control src" and the output of filter 1 is moderated by a gate controlled by the state of "control src". The output of the gate is filtered through "filter 2" and sent to a sink which could be another file on disk or a network socket.

2.1. Software architecture

Prior to 2015, gravitational waves had not been directly observed [3]. Although analysis techniques had been studied for decades [33,34], the character of gravitational wave data evolved with the interferometric detectors during the initial operation of LIGO and Virgo from 2002–2010 [35–40]. Therefore, we aimed to make the GstLAL software modular and easy to adapt to the challenges of Advanced LIGO and Virgo data.

The joint observation of gravitational waves and electromagnetic signals, known as multi-messenger astronomy, was a significant goal for Advanced LIGO and Advanced Virgo [41]. In this scenario, gravitational-wave observations were expected to be followed up by observations with telescopes across the electromagnetic spectrum hoping to catch a short-lived transient light source. Discovering gravitational waves quickly is critical because electromagnetic counterparts may quickly fade. GstLAL was designed to offer analysts an extremely short time-to-solution to help ensure that electromagnetic counterparts could be observed quickly.

The key design principles of GstLAL are:

Plugin-based: Libraries within GstLAL provide Gstreamer plugins to perform gravitational wave specific signal processing tasks. These are mixed together with stock Gstreamer plugins to produce gravitational wave analysis pipelines.

Plugins provide elements, which are the building blocks of signal processing workflows. These elements can be ordered in multiple ways with minimal coding effort which allows for quick exploratory work and development of new methods.

Streaming: The goal of Gstreamer is to provide ultra low latency signal processing suitable for audio and video playback and editing. GstLAL pipelines typically work with streaming data buffers $\lesssim 1$ s in duration.

Event-driven: GstLAL analysis pipelines are designed to run continuously as data are collected. Each application runs an event loop which controls both application level operations as well as settings within a given plugin. This allows for the control of program behavior to be altered while the application is running. Dynamic program control is facilitated through embedding the microservices framework Bottle [42]. Simple http APIs push information to the program, retrieve information or alter the program's behavior.

Scalable: The GstLAL framework is designed both to scale to dozens of cores on a single computer using the multi-threading provided by Gstreamer and to scale to thousands of cores across a compute cluster leveraging HTCondor directed acyclic graph (DAG) scheduling [43].

The GstLAL project is currently comprised of five distinct packages. (1) `gstlal`, (2) `gstlal-ugly`, (3) `gstlal-inspiral`, (4) `gstlal-calibration` and (5) `gstlal-burst`, all of which are described below.

2.2. `gstlal` package

The `gstlal` package curates a set of core plugins, functions and applications that are used by nearly every analysis workflow developed within GstLAL. The `gstlal` package is a dependency for all of the remaining packages which are described in subsequent sections. The `gstlal` package provides Gstreamer elements for Finite-Impulse-Response filtering (`lal_firbank`), $N \rightarrow M$-channel matrix operations (`lal_matrixmixer`), data whitening (`lal_whiten`), and data gating (`lal_gate`). The `gstlal` package also provides basic python APIs for building Gstreamer pipelines in the module `pipeparts`, basic data access routines in the module `datasource`, and a base class for event handling in the module `simplehandler`.

2.3. `gstlal-ugly` package

The `gstlal-ugly` package is an incubator package for software that is in development. Eventually all `gstlal-ugly` software is migrated to one of the other packages.

2.4. `gstlal-inspiral` package

The primary purpose of the `gstlal-inspiral` package is to house the GstLAL-based search for compact binaries [24,25], which centers around the application `gstlal_inspiral`. The GstLAL-based compact binary pipeline was created to make near real-time gravitational-wave detections and aimed to one day detect electromagnetically bright systems before coalescence [44, 45].

The GstLAL-based compact binary search is a matched-filter search that incorporates efficient time-domain filtering [28,44, 46–51] of a set of template waveforms that match the gravitational wave signals of merging black holes and neutron stars [52]. LIGO and Virgo detectors are prone to bursts of nonstationary noise called glitches [19] and determining the difference between gravitational waves and glitches is well suited for many classification algorithms. GstLAL Inspiral implements a classification scheme that is a hybrid of hypothesis testing techniques with some elements coming from machine learning approaches. The classifier is an approximate likelihood ratio comprised of many terms which began as a custom implementation of Naive Bayes classification [53] applied to gravitational wave searches [54]. It was realized early in the project that two things were apparent. First, it was not practical to treat the classifier as entirely data driven relying purely on training sets. Training sets to adequately classify the full parameter space were too expensive to produce. Second, correlations between some parameters had to be tracked in order to classify well. The first point was addressed by developing semi-analytic models to describe parts of the data [55] and the second point was addressed by factoring the multi-dimensional likelihood ratios into groups of lower dimensional, but not one dimensional, distributions [56,57].

The GstLAL-based compact binary search has two modes. The first is a near-real-time, "low-latency" mode that discovers and reports compact binaries within tens of seconds of the signal arriving at Earth. The second is an "offline" mode that efficiently processes data in batch jobs where time-to-solution is not as important as computational efficiency and reproducibility. Although both modes share $\gtrsim 95\%$ of the same code, their behavior and design are very different in order to address the differing concerns of real-time vs. batch processing.

The low-latency mode is a collection of typically $\mathcal{O}(1000)$ microservices that communicate (modestly) with one another asynchronously through http using python Bottle, through an Apache Kafka queue, and through a shared file system. Each one of these microservices processes a portion of the nearly 2 million models used in the current low-latency compact binary search. The low-latency workflow is designed to be fault tolerant. If a job dies, another is restarted to take its place. Since information is exchanged asynchronously and there is no guarantee of job success, the behavior in this mode is non-deterministic. In contrast, the offline mode has a fully deterministic execution which can be reproduced to floating point precision. The determinism is imposed by organizing each job in a directed acyclic graph (DAG) using HTCondor.

2.5. `gstlal-burst` package

The GstLAL-based burst package is a collection of utilities intended to search for gravitational-wave sources other than compact binaries as well as non-astrophysical noise transients. One of the recent developments is a pipeline searching for cosmic strings, which are hypothetical objects considered to have formed in the early universe. The pipeline uses time-domain stream-based signal processing algorithms, along with a classification scheme using parameters specific to the search. The algorithms used are mostly in common with the `gstlal-inspiral` package, but simplified due to the smaller number of templates required for matched filtering.

In addition, `gstlal-burst` provides utilities to identify and extract features from non-Gaussian noise transients in near real-time ($O(5s)$) via the Stream-based Noise Acquisition and eXtraction, or SNAX toolkit [58]. The SNAX toolkit also leverages time-domain signal processing, but instead utilizes a sine-Gaussian basis to identify the presence of and extract features from many types of non-Gaussian noise in strain and auxiliary data. Its main data product is multivariate time-series data containing the extracted features, including SNR and phase information as well as the waveform parameters of interest.

2.6. *gstlal-calibration* package

The `gstlal-calibration` package houses the unique software used for calibration of the LIGO strain data. Software in the `gstlal-calibration` package produces the official LIGO strain data product used in all subsequent analysis. Calibration of LIGO strain data involves standard signal processing and digital filtering techniques in order to derive the differential arm motion observed in the LIGO detectors from the detector's digital readouts [17,59,60]. Many of the signal processing and digital filtering plugins used by the calibration pipeline `gstlal_compute_strain` are housed in the `gstlal` or `gstlal-ugly` packages. As with all other code housed in gstlal-ugly, the relevant pieces of code for calibration currently in gstlal-ugly will eventually get migrated into gstlal or gstlal-calibration, as deemed appropriate. A few plugins unique to the calibration process as well as calibration-specific python APIs are housed in the `gstlal-calibration` package.

Much like the GstLAL-based compact binary pipeline, the LIGO calibration pipeline is built to operate in two modes: a "low-latency" mode and an "offline" mode. The low-latency LIGO calibration pipeline operates on hardware located physically at the two LIGO detector sites, LIGO Hanford in Hanford, WA and LIGO Livingston in Livingston, LA. This pipeline produces calibrated LIGO data and a bit-wise state-vector that indicates the fidelity of the calibrated data within $O(5s)$ for each detector at the respective detector sites. The low-latency calibration process involves using a combination of digital filtering performed in the LIGO front-end computers, which are directly connected to the LIGO detectors and employ the CDS Real-time Code Generator (RCG) core software [61], and further digital filtering and processing performed by the GstLAL calibration software running on non-front-end hardware located at the LIGO detector sites. This two-step process takes advantage of the access the front-end computing system has to the installed detector filters and models and the advanced stream-based filtering techniques housed in the GstLAL software packages [17].

There is often a need to re-calibrate the strain data after the initial low-latency data calibration in order to improve calibration accuracy based on more sophisticated modeling or to remove systematic errors present in the low-latency calibrated strain data [17,62,63]. The re-calibration of LIGO data is performed using the offline mode of the `gstlal_compute_strain` pipeline. In this mode, the entire calibration process is performed by software housed in the GstLAL software packages. The offline calibrated data are processed in batch jobs using HTCondor in order to optimize computational efficiency and is completely reproducible to floating-point precision. All analyses derived from LIGO strain data use the calibrated data produced either by the low-latency GstLAL calibration pipeline or the offline GstLAL calibration pipeline.

3. Illustrative examples

3.1. Example Gstreamer pipeline with GstLAL

The following example was run on a newly instantiated CentOS 7 64 bit virtual machine with miniconda [64] installed by doing:

```
wget https://repo.anaconda.com/miniconda/
    Miniconda3-latest-Linux-x86_64.sh
bash Miniconda3-latest-Linux-x86_64.sh
conda create -n myenv python=2.7
conda activate myenv
conda install -c conda-forge gstlal-
    inspiral=1.7.3
```

To verify that it works, try:

```
$ gst-inspect-1.0 gstlalinspiral | head -n4
Plugin Details:
  Name                        gstlalinspiral
  Description                 Various bits of
    the LIGO Algorithm Library wrapped in
    gstreamer elements
  Filename                    <your miniconda
    path>/lib/gstreamer-1.0/
    libgstgstlalinspiral.so
```

Next we will try a very simple Gstreamer pipeline that uses two GstLAL elements: `lal_peak` and `lal_nxydump`, along with additional Gstreamer elements to construct a pipeline that generates 10 Hz Gaussian, white noise, finds the peak sample every second and streams the result to the terminal screen as ASCII text. It is possible to construct simple pipelines such as this without any code using the Gstreamer tool `gst-launch` [65]:

```
$ gst-launch-1.0 audiotestsrc wave=9 !
    capsfilter caps=audio/x-raw,rate=10 !
    lal_peak n=10 ! lal_nxydump ! filesink
     location=/dev/stdout
```

The first element, `audiotestsrc` is a Gstreamer element that can provide many test signals. The `wave=9` property sets it to be unit variance white noise. The second element, `capsfilter` specifies that we want the format of the output to be floating point audio data with a sample rate of 10 Hz. Next, `lal_peak` is the first GstLAL element. In this example it is configured to find the largest absolute value of the signal every 10 sample points. `lal_nxydump` is the second GstLAL element which converts the time-series data to two column ASCII text. Finally, `filesink` dumps the ASCII output to standard out. You should see the following output (with variations caused by the fact that the data are random):

```
1
2  $ gst-launch-1.0 audiotestsrc wave=9 !
       capsfilter caps=audio/x-raw,rate=10 !
       lal_peak n=10 ! lal_nxydump ! filesink
        location=/dev/stdout | head -n 15
3  Setting pipeline to PAUSED ...
4  Pipeline is PREROLLING ...
5  Pipeline is PREROLLED ...
6  Setting pipeline to PLAYING ...
7  New clock: GstSystemClock
8  0.000000000     0
9  0.100000000     0
10 0.200000000     0
11 0.300000000     0
12 0.400000000     0
13 0.500000000     0
14 0.600000000     0.95523328
15 0.700000000     0
16 0.800000000     0
17 0.900000000     0
```

you can see that the maximum sample point was chosen in the 10 sample interval on line 13. Other values are set to 0.

`gst-launch` is a useful tool for quickly testing a simple pipeline, or debugging, however it is not suitable for writing large applications with many elements or situations where program control is exposed dynamically to the user. For building applications, GstLAL relies on the Python bindings for Gstreamer and adds a substantial amount of gravitational wave specific application code written in python. An example of the pipeline above written in the style of GstLAL applications is below.

```
# boiler plate Gstreamer imports
import gi
gi.require_version('Gst', '1.0')
```

```python
from gi.repository import GObject, Gst
GObject.threads_init()
Gst.init(None)

# Gstlal imports
from gstlal import datasource
from gstlal import pipeparts
from gstlal import simplehandler

# initialize an event loop, a pipeline and an event handler
mainloop = GObject.MainLoop()
pipeline = Gst.Pipeline("softwarex_demo")
handler = simplehandler.Handler(mainloop, pipeline)

src = pipeparts.mkaudiotestsrc(pipeline, wave = 9)
src = pipeparts.mkcapsfilter(pipeline, src, caps = "audio/x-raw, rate=10")
src = pipeparts.mkpeak(pipeline, src, n = 10)
src = pipeparts.mknxydumpsink(pipeline, src, "/dev/stdout")

if pipeline.set_state(Gst.State.PLAYING) == Gst.StateChangeReturn.FAILURE:
    raise RuntimeError("pipeline failed to enter PLAYING state")
mainloop.run()
```

We have found that, using python to procedurally build Gstreamer graphs, we can construct enormous pipelines containing tens of thousands of distinct elements. A prime example of this is our workhorse signal processing pipeline used for discovering compact binary mergers as described in the next section.

3.2. Compact binary searches

`Makefile.softwarex_test` provides an example of the general workflow involved in offline gravitational wave analyses, which primarily rely on the `gstlal` and `gstlal-inspiral` packages. The miniconda installation of GstLAL will run this example in ∼30 min on a single machine. Production level compact binary searches analyzing data from the three advanced interferometers, on the other hand, take ∼1 week when distributed over $\mathcal{O}(1000)$ core computing clusters with optimized software builds. The test Makefile is entirely self-contained; the target and dependency relationships and brief comments within describe the workflow. Although the structure presented in this test is linear in nature, full scale searches for compact binaries are heavily parallelized to take advantage of DAG scheduling. The requisite commands to run this test analysis are shown below.

```bash

mkdir workflow-test && cd workflow-test
export LAL_PATH=${CONDA_PREFIX} GSTLAL_FIR_WHITEN=0 TMPDIR=/tmp
wget https://dcc.ligo.org/public/0168/P2000195/004/Makefile.softwarex_test
make -f Makefile.softwarex_test
```

4. Impact

GstLAL has played an integral role in the history of gravitational wave detections, and has participated in gravitational wave searches since S5 [66]. GstLAL was designed specifically for near-real-time applications, but the low-latency pipeline was only searching for electromagnetically binary neutron stars and neutron star — black hole binaries at the time of GW150914. Nevertheless, GstLAL was one of two matched filter pipelines used to analyze archival data and validate the event [3]. In late 2015 the GstLAL pipeline was approved to include BBHs with total mass $\lesssim 100M_{\odot}$ in its low-latency configuration, and it became the first matched-filter pipeline using waveforms based on general relativity to make a near real-time detection of a compact binary with the discovery of GW151226 [67].

Development work between Advanced LIGO's first and second observing runs focused on enabling single detector discoveries and incorporating data from Virgo in the analysis. These efforts were rewarded by August 2017 as GstLAL made the first three-detector low-latency observation [68] and, more notably, the first (and to date, the only) gravitational-wave detection pipeline to observe a binary neutron star merger in low-latency [69,70]. Although both Advanced LIGO interferometers and the Advanced Virgo interferometer were operating at the time of GW170817, it was initially observed in a single interferometer. This marked the first single detector observation of a gravitational wave, and the autonomous identification of the candidate enabled rapid offline follow-up of the candidate within the LIGO-Virgo collaboration.

Advanced LIGO and Advanced Virgo's third observing run (O3) marked the beginning of open public alerts (OPA) [71]. For the first time, candidates with false-alarm-rates below 1 per month[2] were made public at the time of discovery. Although the first public alert was distributed in error by the collaboration [72], the identification of binary black hole candidate S190408an [73] marked the successful start of the era of automated public alerts. At first only candidates appearing in two or more detectors were approved for automated release, but GW170817 had already demonstrated the importance of single detector searches. Indeed, two weeks into the third observing run GstLAL was the only pipeline to detect GW190425 in near-real-time [69,74], further highlighting the necessity of single detector searches.

Two months into the observing run, GstLAL became the only pipeline approved to release single detector candidates as OPAs. This was a high risk, high reward endeavor. Matched filter searches have traditionally been able to suppress the background by demanding coincidence across interferometers; single detector candidates do not benefit from this effect and can therefore be more susceptible to short duration noise transients. Unfortunately, this resulted in $\mathcal{O}(10)$ retractions throughout O3 as the GstLAL team worked on ways to mitigate the effects of noise transients in single detectors. By the end of the second half of the observing run, pipeline tuning had reduced the rate of retractions. In particular, we found that using a tighter signal model and a linear $h(t)$ gating scheme [24,25] helped avoid spurious candidates.

GstLAL has contributed to all gravitational-wave discoveries published by the LIGO-Virgo Scientific Collaboration [3,8,67,74–83], and it has also contributed to searches for as yet undetected sources. Sub-solar mass pose problems for conventional models of stellar evolution, and GstLAL has directly contributed to searches for both [84,85]. Although these searches have not yet yielded any detections, the null results have been able to place strict limits on the abundance of such objects and have also provided the tightest limit to date on a primordial black hole model of the dark matter.

5. Conclusions

The GstLAL library has significantly impacted the progress of gravitational-wave astrophysics, not only via compact binary searches, but also through contributions to detector calibration and characterization efforts. The low-latency GstLAL based inspiral pipeline was instrumental in the first multi-messenger discovery with gravitational waves, and strives to lead the march towards more remarkable observations with ground-based interferometers.

[2] After applying a trials factor to account for the number of concurrently running searches.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Funding for this work was provided by the National Science Foundation, USA through awards: PHY-1454389, OAC-1642391, PHY-1700765, OAC-1841480, PHY-1607178, and PHY-1847350. Funding for this project was provided by the Charles E. Kaufman Foundation of The Pittsburgh Foundation, USA. Computations for this research were performed on the Pennsylvania State University's Institute for Computational and Data Sciences Advanced CyberInfrastructure (ICDS-ACI) and VM hosting. We are grateful for computational resources provided by the Leonard E Parker Center for Gravitation, Cosmology and Astrophysics at the University of Wisconsin-Milwaukee. Computing support was provided by the LIGO Laboratory through National Science Foundation, USA grant PHY-1764464. GstLAL relies on many other open source software libraries; we gratefully acknowledge the development and support of NumPy [86], SciPy [87], PyGTK [88], PyGST [89], Bottle [42], Kafka [90], Fftw3F [91], Intel MKL [92], GLib2 [93], GNU Scientific Library [94], and GWpy [95]. The authors gratefully acknowledge the LIGO-Virgo-Kagra collaboration, USA for support, review, and valuable critiques throughout various stages of development of the GstLAL library. We are especially thankful for collaborations within the Compact Binary Coalescence working group.

References

[1] Einstein A. Approximative integration of the field equations of gravitation. Sitzungsber Preuss Akad Wiss Berlin (Math Phys) 1916;1916:688–96.
[2] Einstein A. The field equations of gravitation. Sitzungsber Preuss Akad Wiss Berlin (Math Phys) 1915;1915:844–7.
[3] Abbott BP, et al. Observation of gravitational waves from a binary black hole merger. Phys Rev Lett 2016;116(6):061102. http://dx.doi.org/10.1103/PhysRevLett.116.061102, arXiv:1602.03837.
[4] Aasi J, et al., LIGO Scientific Collaboration Advanced LIGO. Classical Quantum Gravity 2015;32:074001. http://dx.doi.org/10.1088/0264-9381/32/7/074001, arXiv:1411.4547.
[5] Acernese F, et al., VIRGO Collaboration Advanced virgo: a second-generation interferometric gravitational wave detector. Classical Quantum Gravity 2015;32(2):024001. http://dx.doi.org/10.1088/0264-9381/32/2/024001, arXiv:1408.3978.
[6] Somiya K, KAGRA Collaboration. Detector configuration of KAGRA: The Japanese cryogenic gravitational-wave detector. In: Hannam M, Sutton P, Hild S, van den Broeck C, editors. Classical Quantum Gravity 2012;29:124007. http://dx.doi.org/10.1088/0264-9381/29/12/124007, arXiv:1111.7185.
[7] Aso Y, Michimura Y, Somiya K, Ando M, Miyakawa O, Sekiguchi T, et al., KAGRA Collaboration Interferometer design of the KAGRA gravitational wave detector. Phys Rev D 2013;88(4):043007. http://dx.doi.org/10.1103/PhysRevD.88.043007, arXiv:1306.6747.
[8] Abbott BP, et al., LIGO Scientific Virgo Collaboration GWTC-1: A gravitational-wave transient catalog of compact binary mergers observed by LIGO and virgo during the first and second observing runs. Phys Rev X 2019;9(3):031040. http://dx.doi.org/10.1103/PhysRevX.9.031040, arXiv:1811.12907.
[9] Abbott BP, et al., LIGO Scientific Virgo Collaboration Binary black hole population properties inferred from the first and second observing runs of advanced LIGO and advanced virgo. Astrophys J 2019;882(2):L24. http://dx.doi.org/10.3847/2041-8213/ab3800, arXiv:1811.12940.
[10] Abbott BP, et al., LIGO Scientific Virgo, 1M2H, Dark Energy Camera GW-E, DES, DLT40, Las Cumbres Observatory, VINROUGE, MASTER Collaboration A gravitational-wave standard siren measurement of the hubble constant. Nature 2017;551(7678):85–8. http://dx.doi.org/10.1038/nature24471, arXiv:1710.05835.
[11] Abbott BP, et al., LIGO Scientific Virgo Collaboration Tests of general relativity with GW170817. Phys Rev Lett 2019;123(1):011102. http://dx.doi.org/10.1103/PhysRevLett.123.011102, arXiv:1811.00364.
[12] Abbott BP, et al., LIGO Scientific Virgo Collaboration Tests of general relativity with the binary black hole signals from the LIGO-virgo catalog GWTC-1. Phys Rev D 2019;100(10):104036. http://dx.doi.org/10.1103/PhysRevD.100.104036, arXiv:1903.04467.
[13] Abbott B, Abbott R, Abbott T, Abraham S, Acernese F, Ackley K, et al. A gravitational-wave measurement of the hubble constant following the second observing run of advanced LIGO and virgo. 2019, arXiv preprint arXiv:1908.06060.
[14] Abbott BP, et al., LIGO Scientific Virgo, Fermi GBM, INTEGRAL, IceCube, AstroSat Cadmium Zinc Telluride Imager Team, IPN, Insight-Hxmt, ANTARES, Swift, AGILE Team, 1M2H Team, Dark Energy Camera GW-EM, DES, DLT40, GRAWITA, Fermi-LAT, ATCA, ASKAP, Las Cumbres Observatory Group, OzGrav, DWF (Deeper Wider Faster Program), AST3, CAASTRO, VINROUGE, MASTER, J-GEM, GROWTH, JAGWAR, CaltechNRAO, TTU-NRAO, NuSTAR, Pan-STARRS, MAXI Team, TZAC Consortium, KU, Nordic Optical Telescope, ePESSTO, GROND, Texas Tech University, SALT Group, TOROS, BOOTES, MWA, CALET, IKI-GW Follow-up, HESS, LOFAR, LWA, HAWC, Pierre Auger, ALMA, Euro VLBI Team, Pi of Sky, Chandra Team at McGill University, DFN, ATLAS Telescopes, High Time Resolution Universe Survey, RIMAS, RATIR, SKA South Africa/MeerKAT Collaboration Multi-messenger observations of a binary neutron star merger. Astrophys J 2017;848(2):L12. http://dx.doi.org/10.3847/2041-8213/aa91c9, arXiv:1710.05833.
[15] Abbott BP, et al., LIGO Scientific Virgo Collaboration GW170817: Measurements of neutron star radii and equation of state. Phys Rev Lett 2018;121(16):161101. http://dx.doi.org/10.1103/PhysRevLett.121.161101, arXiv:1805.11581.
[16] Nicholl M, et al. The electromagnetic counterpart of the binary neutron star merger LIGO/VIRGO GW170817. III. Optical and UV spectra of a blue kilonova from fast polar ejecta. Astrophys J 2017;848(2):L18. http://dx.doi.org/10.3847/2041-8213/aa9029, arXiv:1710.05456.
[17] Viets A, et al. Reconstructing the calibrated strain signal in the Advanced LIGO detectors. Classical Quantum Gravity 2018;35(9):095015. http://dx.doi.org/10.1088/1361-6382/aab658, arXiv:1710.09973.
[18] Acernese F, et al., Virgo Collaboration Calibration of advanced virgo and reconstruction of the gravitational wave signal $h(t)$ during the observing run O2. Classical Quantum Gravity 2018;35(20):205004. http://dx.doi.org/10.1088/1361-6382/aadf1a, arXiv:1807.03275.
[19] Abbott BP, et al., LIGO Scientific Virgo Collaboration Characterization of transient noise in advanced LIGO relevant to gravitational wave signal GW150914. Classical Quantum Gravity 2016;33(13):134001. http://dx.doi.org/10.1088/0264-9381/33/13/134001, arXiv:1602.03844.
[20] Science Instrument List L, Buikema A, Cahillane C, Mansell GL, Blair CD, Abbott R, et al. Sensitivity and performance of the advanced LIGO detectors in the third observing run. 2020, arXiv e-prints arXiv:2008.01301.
[21] GstLAL software:.
[22] Essick R, Godwin P, Hanna C, Blackburn L, Katsavounidis E. IDQ: Statistical inference of non-Gaussian noise with auxiliary degrees of freedom in gravitational-wave detectors. 2020, arXiv:2005.12761.
[23] Godwin P, et al. Incorporation of statistical data quality information into the gstlal search analysis. 2020, arXiv:2010.15282.
[24] Messick C, et al. Analysis framework for the prompt discovery of compact binary mergers in gravitational-wave data. Phys Rev 2017;D95(4):042001. http://dx.doi.org/10.1103/PhysRevD.95.042001, arXiv:1604.04324.
[25] Sachdev S, et al. The GstLAL search analysis methods for compact binary mergers in advanced LIGO's second and advanced virgo's first observing runs. 2019, arXiv:1901.08580.
[26] Olson H. Music, physics and engineering. Dover books, Dover Publications; 1967, https://books.google.com/books?id=RUDTFBbb7jAC.
[27] Gstreamer : https://gstreamer.freedesktop.org/.
[28] Cannon K, Chapman A, Hanna C, Keppel D, Searle AC, Weinstein AJ. Singular value decomposition applied to compact binary coalescence gravitational-wave signals. Phys Rev 2010;D82:044025. http://dx.doi.org/10.1103/PhysRevD.82.044025, arXiv:1005.0012.
[29] LIGO Software: http://software.ligo.org.
[30] LIGO Packages: http://software.igwn.org/lscsoft/.
[31] Container Registry - lscsoft/gstlal: https://git.ligo.org/lscsoft/gstlal/container_registry.
[32] Gstlal Inspiral :: Anaconda Cloud: https://anaconda.org/conda-forge/gstlal-inspiral.
[33] Finn LS, Chernoff DF. Observing binary inspiral in gravitational radiation: One interferometer. Phys Rev 1993;D47:2198–219. http://dx.doi.org/10.1103/PhysRevD.47.2198, arXiv:gr-qc/9301003.
[34] Allen B, Anderson WG, Brady PR, Brown DA, Creighton JDE. FINDCHIRP: An algorithm for detection of gravitational waves from inspiraling compact binaries. Phys Rev 2012;D85:122006. http://dx.doi.org/10.1103/PhysRevD.85.122006, arXiv:gr-qc/0509116.
[35] Abbott B, et al., LIGO Scientific Collaboration Analysis of LIGO data for gravitational waves from binary neutron stars. Phys Rev D 2004;69:122001. http://dx.doi.org/10.1103/PhysRevD.69.122001, arXiv:gr-qc/0308069.

[36] Abbott B, et al., LIGO Scientific Collaboration Search for gravitational waves from galactic and extra-galactic binary neutron stars. Phys Rev D 2005;72:082001. http://dx.doi.org/10.1103/PhysRevD.72.082001, arXiv:gr-qc/0505041.

[37] Abbott B, et al., LIGO Scientific Collaboration Search for gravitational waves from binary inspirals in S3 and S4 LIGO data. Phys Rev D 2008;77:062002. http://dx.doi.org/10.1103/PhysRevD.77.062002, arXiv:0704.3368.

[38] Abbott BP, et al., LIGO Scientific Collaboration Search for gravitational waves from low mass binary coalescences in the first year of LIGO's S5 data. Phys Rev D 2009;79:122001. http://dx.doi.org/10.1103/PhysRevD.79.122001, arXiv:0901.0302.

[39] Abbott BP, et al., LIGO Scientific Collaboration Search for gravitational waves from low mass compact binary coalescence in 186 days of LIGO's fifth science run. Phys Rev D 2009;80:047101. http://dx.doi.org/10.1103/PhysRevD.80.047101, arXiv:0905.3710.

[40] Abadie J, et al., LIGO Scientific VIRGO Collaboration Search for gravitational waves from low mass compact binary coalescence in LIGO's sixth science run and virgo's science runs 2 and 3. Phys Rev D 2012;85:082002. http://dx.doi.org/10.1103/PhysRevD.85.082002, arXiv:1111.7314.

[41] Kanner J, Huard T, Márka S, Murphy D, Piscionere J, Reed M, et al. LOOC UP: Locating and observing optical counterparts to gravitational wave bursts. In: Hughes S, Katsavounidis E, editors. Classical Quantum Gravity 2008;25:184034. http://dx.doi.org/10.1088/0264-9381/25/18/184034, arXiv:0803.0312.

[42] Bottle: Python Web Framework: https://bottlepy.org/docs/dev/.

[43] HTCondor High Throughput Computing: https://research.cs.wisc.edu/htcondor/.

[44] Cannon K, et al. Toward early-warning detection of gravitational waves from compact binary coalescence. Astrophys J 2012;748:136. http://dx.doi.org/10.1088/0004-637X/748/2/136, arXiv:1107.2665.

[45] Sachdev S, et al. An early warning system for electromagnetic follow-up of gravitational-wave events. 2020, arXiv:2008.04288.

[46] Cannon K, Hanna C, Keppel D, Searle AC. Composite gravitational-wave detection of compact binary coalescence. Phys Rev D 2011;83:084053. http://dx.doi.org/10.1103/PhysRevD.83.084053, arXiv:1101.0584.

[47] Cannon K, Hanna C, Keppel D. Efficiently enclosing the compact binary parameter space by singular-value decomposition. Phys Rev D 2011;84:084003. http://dx.doi.org/10.1103/PhysRevD.84.084003, arXiv:1101.4939.

[48] Cannon K, Hanna C, Keppel D. Interpolating compact binary waveforms using the singular value decomposition. In: Hannam M, Sutton P, Hild S, van den Broeck C, editors. Phys Rev D 2012;85:081504. http://dx.doi.org/10.1103/PhysRevD.85.081504, arXiv:1108.5618.

[49] Cannon K, Emberson J, Hanna C, Keppel D, Pfeiffer H. Interpolation in waveform space: enhancing the accuracy of gravitational waveform families using numerical relativity. Phys Rev D 2013;87(4):044008. http://dx.doi.org/10.1103/PhysRevD.87.044008, arXiv:1211.7095.

[50] Smith R, Cannon K, Hanna C, Keppel D, Mandel I. Towards rapid parameter estimation on gravitational waves from compact binaries using interpolated waveforms. Phys Rev D 2013;87(12):122002. http://dx.doi.org/10.1103/PhysRevD.87.122002, arXiv:1211.1254.

[51] Tsukada L, Cannon K, Hanna C, Keppel D, Meacher D, Messick C. Application of a zero-latency whitening filter to compact binary coalescence gravitational-wave searches. Phys Rev D 2018;97(10):103009. http://dx.doi.org/10.1103/PhysRevD.97.103009, arXiv:1708.04125.

[52] Mukherjee D, et al. The GstLAL template bank for spinning compact binary mergers in the second observation run of advanced LIGO and virgo. 2018, arXiv:1812.05121.

[53] Zhang H. The optimality of naive Bayes. AA 2004;1(2):3.

[54] Cannon KC. A Bayesian coincidence test for noise rejection in a gravitational-wave burst search. Classical Quantum Gravity 2008;25:105024. http://dx.doi.org/10.1088/0264-9381/25/10/105024.

[55] Hanna C, et al. Fast evaluation of multi-detector consistency for real-time gravitational wave searches. Phys Rev 2020;D101(2):022003. http://dx.doi.org/10.1103/PhysRevD.101.022003, arXiv:1901.02227.

[56] Cannon K, Hanna C, Peoples J. Likelihood-ratio ranking statistic for compact binary coalescence candidates with rate estimation. 2015, arXiv:1504.04632.

[57] Cannon K, Hanna C, Keppel D. Method to estimate the significance of coincident gravitational-wave observations from compact binary coalescence. Phys Rev 2013;D88(2):024025. http://dx.doi.org/10.1103/PhysRevD.88.024025, arXiv:1209.0718.

[58] Godwin P. Low-latency statistical data quality in the era of multi-messenger astronomy (Ph.D. thesis), Penn State U.; 2020.

[59] Abbott BP, et al., LIGO Scientific Collaboration Calibration of the advanced LIGO detectors for the discovery of the binary black-hole merger GW150914. Phys Rev D 2017;95(6):062003. http://dx.doi.org/10.1103/PhysRevD.95.062003, arXiv:1602.03845.

[60] Tuyenbayev D, Karki S, Betzwieser J, Cahillane C, Goetz E, Izumi K, et al. Improving LIGO calibration accuracy by tracking and compensating for slow temporal variations. Classical Quantum Gravity 2016;34(1):015002. http://dx.doi.org/10.1088/0264-9381/34/1/015002.

[61] Bork R, Thorne K, Barker D, Hanks J. Real-time code generator (RCG) version 3.0 release notes. T1600055, 2016.

[62] Cahillane C, Betzwieser J, Brown DA, Goetz E, Hall ED, Izumi K, et al. Calibration uncertainty for advanced LIGO's first and second observing runs. Phys Rev D 2017;96:102001. http://dx.doi.org/10.1103/PhysRevD.96.102001, https://link.aps.org/doi/10.1103/PhysRevD.96.102001.

[63] Sun L, Goetz E, Kissel JS, Betzwieser J, Karki S, Viets A, et al. Characterization of systematic error in advanced LIGO calibration. 2020, arXiv e-prints arXiv:2005.02531.

[64] Miniconda: https://docs.conda.io/en/latest/miniconda.html.

[65] gst-launch-1.0: https://gstreamer.freedesktop.org/documentation/tools/gst-launch.html?gi-language=c.

[66] Wade M. Gravitational-wave science with the laser interferometer gravitational-wave observatory (Ph.D. thesis), University of Wisconsin-Milwaukee; 2015, arXiv:LIGO-P1500068, https://dcc.ligo.org/LIGO-P1500068/public.

[67] Abbott BP, et al., LIGO Scientific Virgo Collaboration GW151226: Observation of gravitational waves from a 22-solar-mass binary black hole coalescence. Phys Rev Lett 2016;116(24):241103. http://dx.doi.org/10.1103/PhysRevLett.116.241103, arXiv:1606.04855.

[68] LIGO Scientific Collaboration VC. GCN 21474. 2017, https://gcn.gsfc.nasa.gov/gcn3/21474.gcn3.

[69] LIGO Scientific Collaboration VC. GCN 24168. 2019, https://gcn.gsfc.nasa.gov/gcn3/24168.gcn3.

[70] LIGO Scientific Collaboration VC. GCN g298048. 2017, https://gcn.gsfc.nasa.gov/other/G298048.gcn3.

[71] LIGO Scientific Collaboration VC. GCN 24045. 2019, https://gcn.gsfc.nasa.gov/gcn3/24045.gcn3.

[72] LIGO Scientific Collaboration VC. GCN 24109. 2019, https://gcn.gsfc.nasa.gov/gcn3/24109.gcn3.

[73] LIGO Scientific Collaboration VC. Gcn 24069. 2019, https://gcn.gsfc.nasa.gov/gcn3/24069.gcn3.

[74] Abbott BP, et al., LIGO Scientific Virgo Collaboration GW190425: Observation of a compact binary coalescence with total mass $\sim 3.4M_{\odot}$. 2020, arXiv:2001.01761.

[75] Abbott R, et al. GWTC-2: Compact binary coalescences observed by LIGO and virgo during the first half of the third observing run. 2020, arXiv:2010.14527.

[76] Abbott R, et al., LIGO Scientific Virgo Collaboration Population properties of compact objects from the second LIGO-virgo gravitational-wave transient catalog. 2020, arXiv:2010.14533.

[77] Abbott R, et al., LIGO Scientific Virgo Collaboration GW190521: A binary black hole merger with a total mass of $150M_{\odot}$. Phys Rev Lett 2020;125(10):101102. http://dx.doi.org/10.1103/PhysRevLett.125.101102, arXiv:2009.01075.

[78] Abbott R, et al., LIGO Scientific Virgo Collaboration GW190814: Gravitational waves from the coalescence of a 23 solar mass black hole with a 2.6 solar mass compact object. Astrophys J Lett 2020;896(2):L44. http://dx.doi.org/10.3847/2041-8213/ab960f, arXiv:2006.12611.

[79] Abbott R, et al., LIGO Scientific Virgo Collaboration GW190412: Observation of a binary-black-hole coalescence with asymmetric masses. Phys Rev D 2020;102(4):043015. http://dx.doi.org/10.1103/PhysRevD.102.043015, arXiv:2004.08342.

[80] Abbott BP, et al., LIGO Scientific Virgo Collaboration GW170817: Observation of gravitational waves from a binary neutron star inspiral. Phys Rev Lett 2017;119(16):161101. http://dx.doi.org/10.1103/PhysRevLett.119.161101, arXiv:1710.05832.

[81] Abbott BP, et al., LIGO Scientific VC GW170814: A three-detector observation of gravitational waves from a binary black hole coalescence. Phys Rev Lett 2017;119(14):141101. http://dx.doi.org/10.1103/PhysRevLett.119.141101, arXiv:1709.09660.

[82] Abbott BP, et al., LIGO Scientific VC GW170608: Observation of a 19-solar-mass binary black hole coalescence. Astrophys J 2017;851(2):L35. http://dx.doi.org/10.3847/2041-8213/aa9f0c, arXiv:1711.05578.

[83] Abbott BP, et al., LIGO Scientific VC GW170104: Observation of a 50-solar-mass binary black hole coalescence at redshift 0.2. Phys Rev Lett 2017;118(22):221101. http://dx.doi.org/10.1103/PhysRevLett.118.221101, arXiv:1706.01812.

[84] Abbott BP, et al., LIGO Scientific VC Search for subsolar-mass ultracompact binaries in advanced LIGO's first observing run. Phys Rev Lett 2018;121(23):231103. http://dx.doi.org/10.1103/PhysRevLett.121.231103, arXiv:1808.04771.

[85] Abbott BP, et al., LIGO Scientific VC Search for subsolar mass ultracompact binaries in advanced LIGO's second observing run. Phys Rev Lett 2019;123(16):161102. http://dx.doi.org/10.1103/PhysRevLett.123.161102, arXiv:1904.08976.

[86] Oliphant T. NumPy: A guide to NumPy. USA: Trelgol Publishing; 2006, [Online; accessed <today>] http://www.numpy.org/.
[87] Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, et al. Scipy 1.0: Fundamental algorithms for scientific computing in python. Nat Methods 2020;17:261–72. http://dx.doi.org/10.1038/s41592-019-0686-2.
[88] PyGtk software: https://wiki.python.org/moin/PyGtk.
[89] Erik, Saldyt L, Rob, Gross J, tjproct, kmrudin, et al. Pygstio/pygsti: Version 0.9.9.1. 2020, http://dx.doi.org/10.5281/zenodo.3675466.
[90] Apache Kafka software: https://kafka.apache.org/.
[91] Frigo M, Johnson SG. The design and implementation of FFTW3. Proc IEEE 2005;93(2):216–31.
[92] Intel Math Kernel Library: https://software.intel.com/content/www/us/en/develop/tools/math-kernel-library.html.
[93] GLib2 software: https://github.com/GNOME/glib.
[94] Galassi M, Davies J, Theiler J, Gough B, Jungman G. GNU scientific library - reference manual. 3rd ed.. 2009, for GSL Version 1.12 (3. ed.).
[95] Macleod D, Urban AL, Coughlin S, Massinger T, Pitkin M, paulaltin, et al. Gwpy/gwpy: 1.0.1. 2020, http://dx.doi.org/10.5281/zenodo.3598469.