DCC: LIGO-G2300986

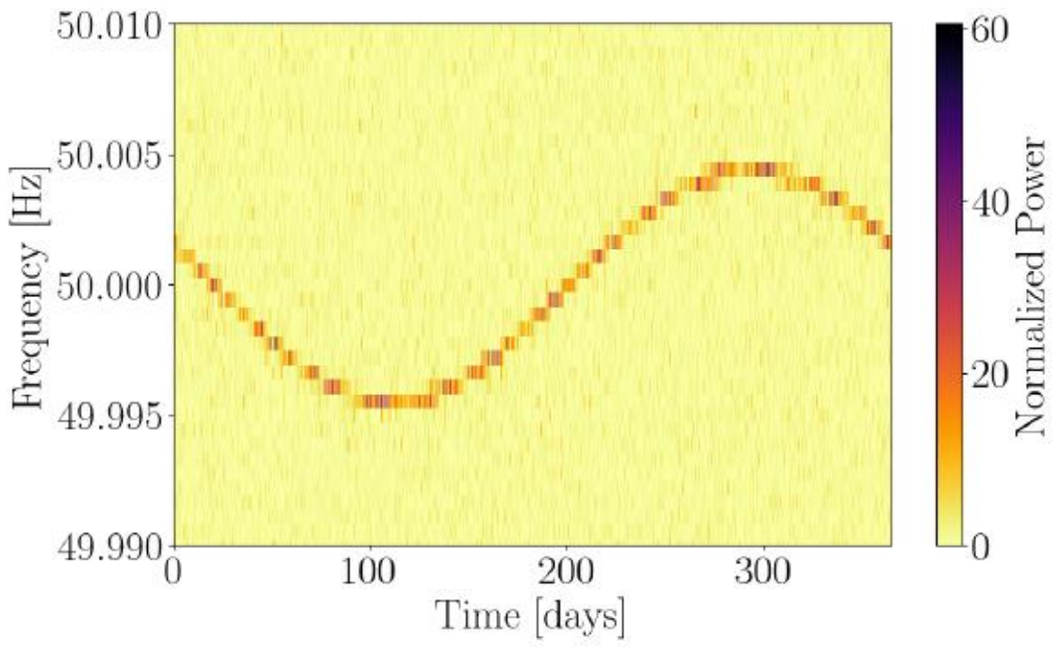
Tutorial: *F*-statistic analyses with lalpulsar and PyFstat

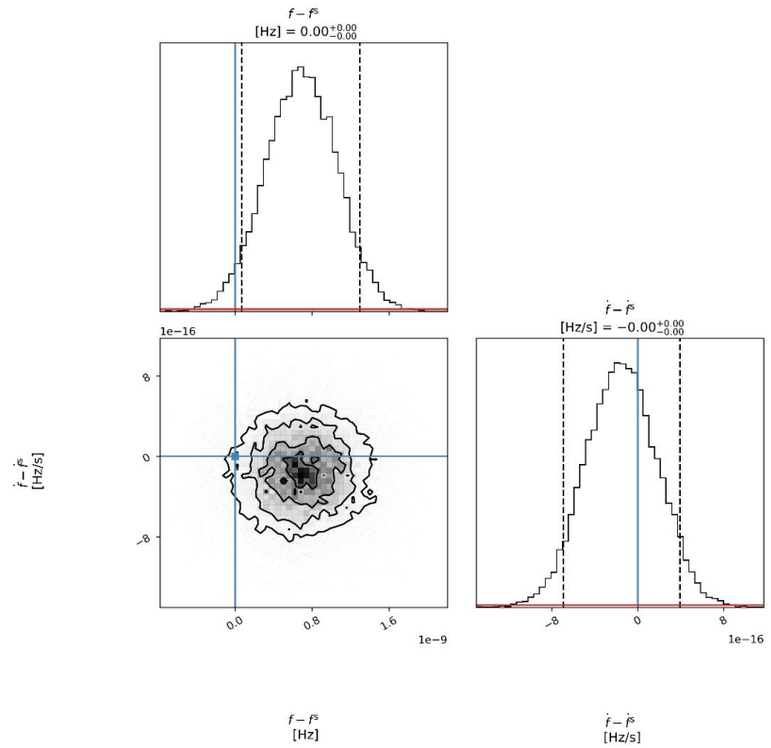
David Keitel & Rodrigo Tenorio

(IAC3 / Universitat de les Illes Balears)

Universitat de les Illes Balears

IAC3 Institute of Applied Computing & Community Code.





CW matched filter & the *F*-statistic

noise and signal likelihoods, hypothesis testing:

- likelihood of measuring data x under Gaussian noise hypothesis:

$$\mathcal{H}_{\mathrm{G}} : \boldsymbol{x}(t) = \boldsymbol{n}(t) \longrightarrow P\left(\boldsymbol{x}|\mathcal{H}_{\mathrm{G}}\right) = \kappa\, \mathrm{e}^{-\frac{1}{2}\langle \boldsymbol{x}|\boldsymbol{x}\rangle}$$

- (composite) signal hypothesis: $\mathcal{H}_{\mathrm{S}} : \boldsymbol{x}(t) = \boldsymbol{n}(t) + \boldsymbol{h}(t;\mathcal{A},\lambda)$

$$\longrightarrow P\left(\boldsymbol{x}|\mathcal{H}_{\mathrm{S}},\mathcal{A}\right) = \kappa\, \mathrm{e}^{-\frac{1}{2}\langle \boldsymbol{x}-\boldsymbol{h}(\mathcal{A})|\boldsymbol{x}-\boldsymbol{h}(\mathcal{A})\rangle}$$
$$= P\left(\boldsymbol{x}|\mathcal{H}_{\mathrm{G}}\right)\, \mathrm{e}^{\mathcal{A}^\mu\langle \boldsymbol{x}|\boldsymbol{h}_\mu\rangle - \frac{1}{2}\mathcal{A}^\mu\langle \boldsymbol{h}_\mu|\boldsymbol{h}_\nu\rangle\mathcal{A}^\nu}$$

- (log) likelihood ratio (matched-filter statistic): $\log\Lambda(x) = \langle x\,|s\rangle - \frac{1}{2}\langle s\,|s\rangle$
- *Maximum-likelihood* estimator of the *amplitude* parameters:

$$\frac{\partial \log\Lambda(\boldsymbol{x},\mathcal{A},\lambda)}{\partial\mathcal{A}^\mu} = x_\mu(\lambda) - \mathcal{M}_{\mu\nu}(\lambda)\,\mathcal{A}^\nu \stackrel{!}{=} 0 \longrightarrow \boxed{2\mathcal{F}(\boldsymbol{x},\lambda) \equiv x_\mu\,\mathcal{M}^{\mu\nu}(\lambda)\,x_\nu}$$

- Alternative Bayesian derivation: *marginalise* over a certain amplitude prior choice [Prix&Krishnan2009]

$$O_{\mathrm{SG}}(\boldsymbol{x}) \sim \boxed{B_{\mathrm{SG}}(\boldsymbol{x}) \equiv \frac{P(\boldsymbol{x}|\mathcal{H}_{\mathrm{S}})}{P(\boldsymbol{x}|\mathcal{H}_{\mathrm{G}})} = c_*^{-1}\mathrm{e}^{\mathcal{F}(\boldsymbol{x})}}$$

SFTs

- From Tuesday talk: “Data is usually split up into Short Fourier Transforms (SFTs), typically of T_{SFT}=1800s or similar.”
- Used by most LIGO/LVC/LVK CW searches, can be done for all detectors. (But some Virgo-developed pipelines use different formats that are also Fourier transforms of short duration, but not in the SFT file format: “SFDBs”)
- Format specification: https://dcc.ligo.org/T040164/public
- Normally no-one ever reads this, but updated in 2022 to v3:
  - window information now stored in header (using old padding bytes)
  - file names: non-alphanumeric characters (e.g. underscores!) no longer allowed
- A simple pure LALSuite example:

```
mkdir test_SFTs
lalpulsar_Makefakedata_v5 --IFO=L1 --sqrtSX=1e-23
--fmin=95 --Band=10 --startTime=1320000000 --
duration=259200 --outSFTdir=test_SFTs --
outSingleSFT=False
```

- The last added options make this produce 144 *single-timestamp* SFT files:

```
L-1_L1_1800SFT_mfdv5-1320000000-1800.sft
L-1_L1_1800SFT_mfdv5-1320001800-1800.sft      ...and so on
L-1_L1_1800SFT_mfdv5-1320003600-1800.sft
```

SFTs

- SFTs are binary files, but their content is easy to check with a certain tool:

```
lalpulsar_dumpSFT --SFTfiles test_SFTs/L-1_L1_1800SFT_mfdv5-1320000000-1800.sft > less

%% ----- Header:
Name:        'L1'
epoch:       [1320000000, 0]
f0:          95.000000000
deltaF:      0.000555555556
numBins:     18000
%% ----- Descriptor:
Locator:     'test_SFTs/L-1_L1_1800SFT_mfdv5-1320000000-1800.sft : 0'
SFT version: 3
numBins:     18000
crc64:       16538776058293058864
window:      rectangular(0)
comment: ==========
L1
Generated by:
/home/dkeitel/bin/venvs/pyfstat/bin/lalpulsar_Makefakedata_v5 --outSingleSFT=FALSE --
outSFTdir="test_SFTs" --IFOs="L1" --sqrtSX="1e-23" --startTime=1320000000 --duration=259200 --
fmin=95 --Band=10
[…]
end comment: ------
%% ----- Data x(f):
%% Frequency f[Hz]     Real(x)         Imag(x)
95.000000000      -2.740061e-22   0.000000e+00
95.000555556       8.339156e-23  -9.098723e-23
[…]
```

SFTs

- dump data only (no header, `-d` flag):

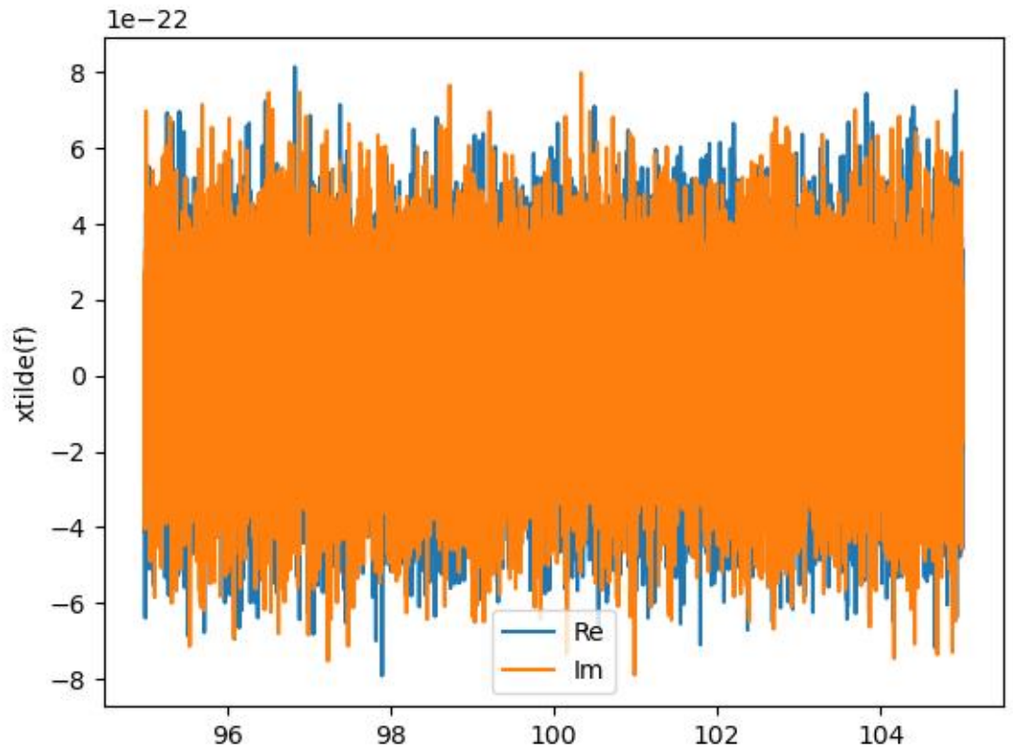
```
lalpulsar_dumpSFT --SFTfiles test_SFTs/L-
1_L1_1800SFT_mfdv5-1320000000-1800.sft -d >
test_SFTs/L-1_L1_1800SFT_mfdv5-1320000000-
1800.sft.dump
```

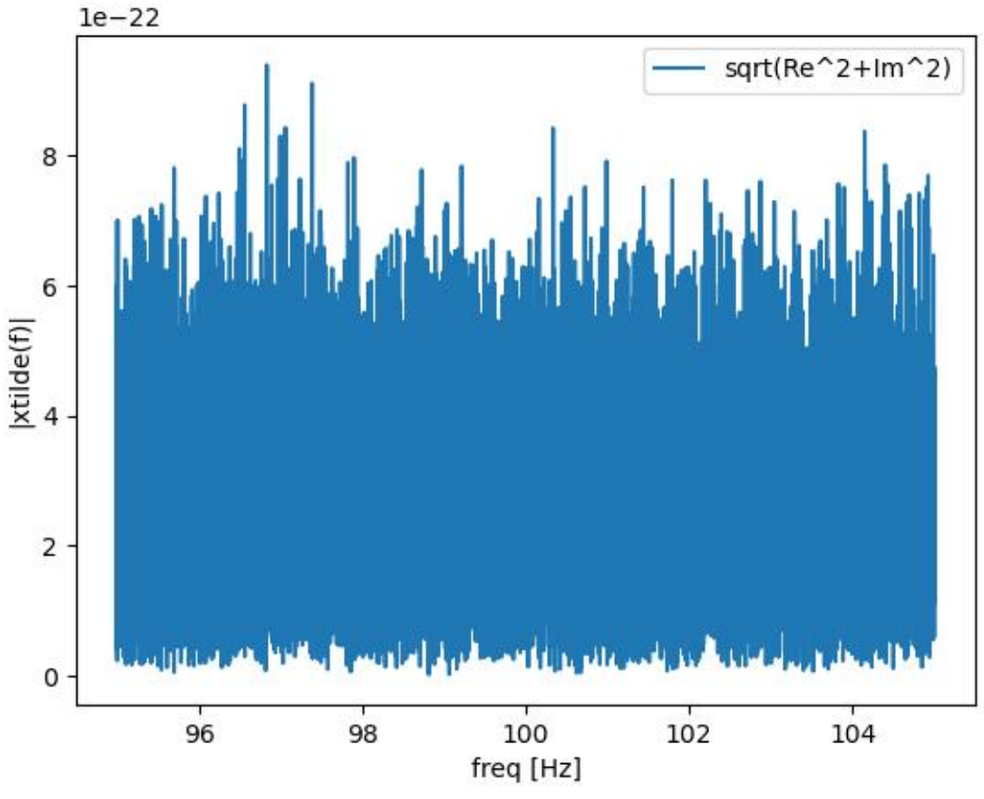
- plotting the raw data with python:

```
import numpy as np
import matplotlib.pyplot as plt
sft = np.genfromtxt("test_SFTs/L-
1_L1_1800SFT_mfdv5-1320000000-
1800.sft.dump", comments="%%")
plt.plot(sft[:,0], sft[:,1],
label="Re")
plt.plot(sft[:,0], sft[:,2],
label="Im")
plt.show()
```

- absolute value (“SFT power”):

```
plt.plot(sft[:,0],
np.sqrt(sft[:,1]**2+sft[:,2]**2),
label="sqrt(Re^2+Im^2)")
plt.show()
```

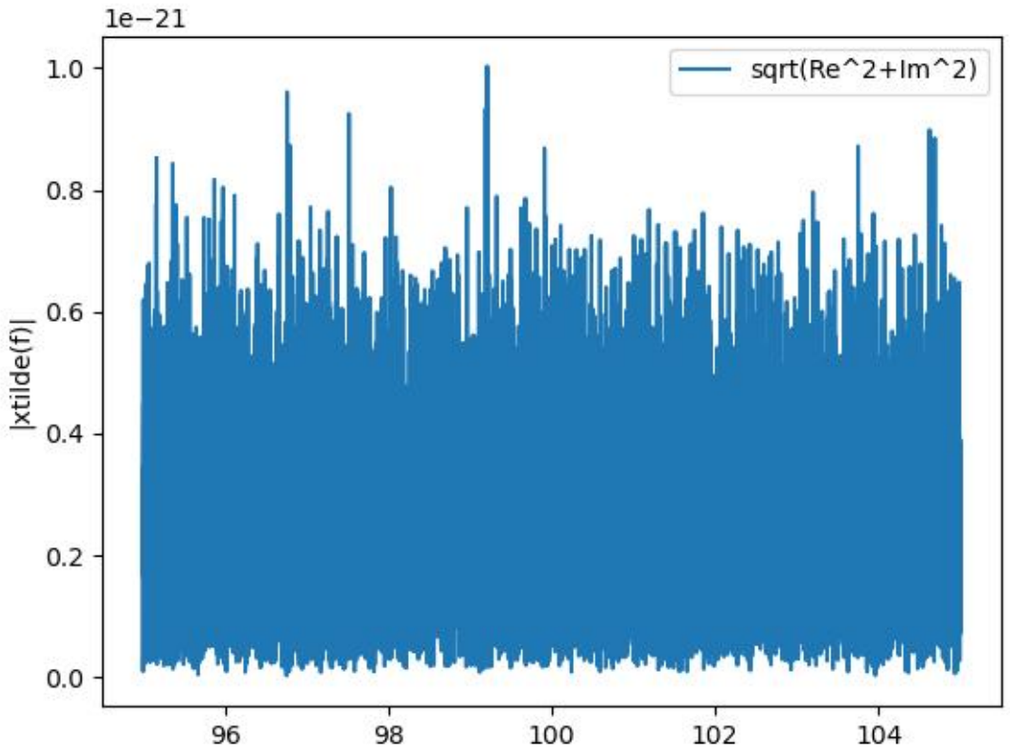




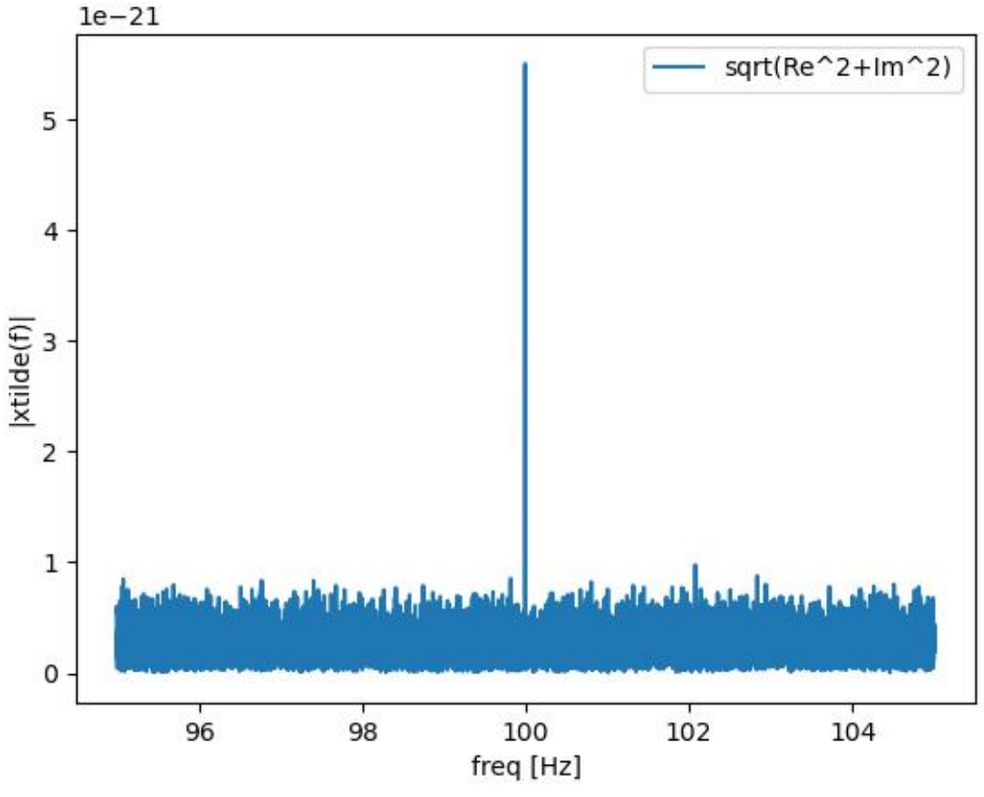
SFTs

- With a signal in it:

```
mkdir test_SFTs_inj
lalpulsar_Makefakedata_v5 --
IFO=L1 --sqrtSX=1e-23 --fmin=95
--Band=10 --startTime=1320000000
--duration=259200 --
outSFTdir=test_SFTs_inj --
outSingleSFT=False --
injectionSources="{Alpha=0;
Delta=0; Freq=100; f1dot=-5e-10;
f2dot=0; refTime=1320000000;
h0=5e-25; cosi=0.1; psi=0;
phi0=0}"
```



- Doesn't look like much?
  Let's try a much larger h0=5e-21:



- We need such a huge h0 because we're looking at the raw data in a single SFT. This is obviously not what a real CW search will do!

SFTs

- Without “`--outSingleSFT=False`”,

```
lalpulsar_Makefakedata_v5 --IFO=L1 --sqrtSX=1e-23
--fmin=95 --Band=10 --startTime=1320000000 --
duration=259200 --outSFTdir=test_SFTs
```

  we get the more convenient “merged SFTs” format:
  the file `L-144_L1_1800SFT_mfdv5-1320000000-259200.sft` contains all 144 SFTs of length 1800s covering the requested duration of 3 days = 259200.

- When dumped, this will just be equivalent to dumping all the individual files and🐈-ing them.

- These test SFTs we’ve generated were also *narrow-banded.*

- When working with real data, the starting point (after conversion from .gwf frames) are usually single-timestamp (unmerged), broad-band (2kHz or more) files.

- For practical searches, these are repartitioned with tools like `lalpulsar_SplitSFTs`.

PSDs

- In CBC land, PSD estimation is a big challenge because one needs to catch the exact behaviour of the detectors near a short-duration transient signal.

- For CWs on the other hand, we care about long-duration average detector behaviour:

$$\widehat{S}^X(f') \equiv \frac{1}{N^X_{\mathrm{SFT}}} \sum_{\alpha=1}^{N^X_{\mathrm{SFT}}} \frac{2\left|\tilde{x}^X_\alpha(f')\right|^2}{T_{\mathrm{SFT}}}$$

where e.g. for a 1-year run, Tsft=1800s and 60% duty factor, we'd have ~10k SFTs to average over.

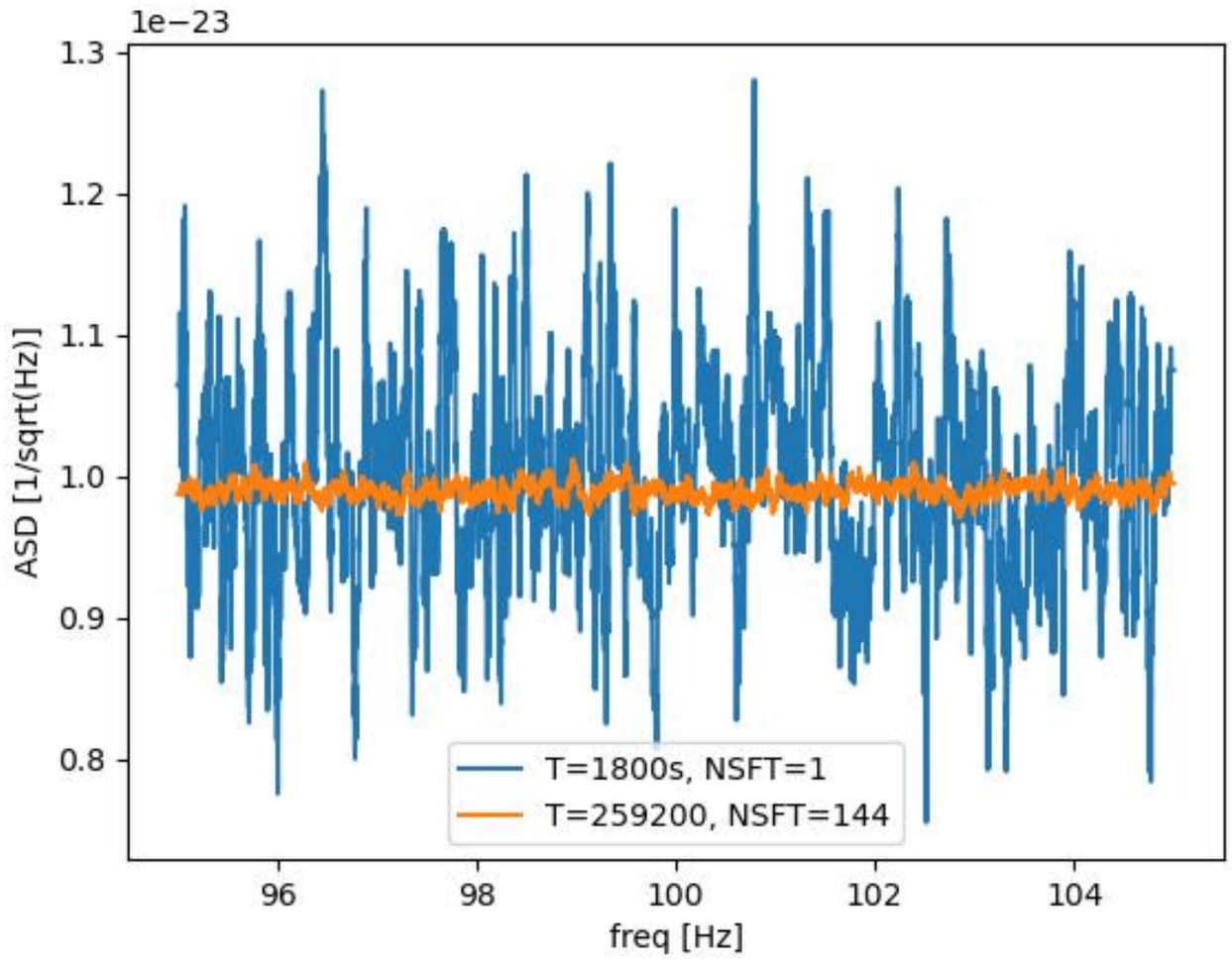
- Let's compare PSDs over 1800s and over 3 days:

```
lalpulsar_ComputePSD --inputData=test_SFTs/L-
1_L1_1800SFT_mfdv5-1320000000-1800.sft --
outputPSD=test_SFTs/L-1_L1_1800SFT_mfdv5-1320000000-
1800.sft.psd
lalpulsar_ComputePSD --inputData=test_SFTs/L-
144_L1_1800SFT_mfdv5-1320000000-259200.sft --
outputPSD=test_SFTs/L-144_L1_1800SFT_mfdv5-1320000000-
259200.sft.psd
```

PSDs

- `lalpulsar_ComputePSD` output files are just 2-column text files with the same 1/Tsft bin size as the original SFT:

```
[...]
%% FreqBinStart PSD
95.000000   9.757549e-47
95.000556   9.757549e-47
95.001111   9.757549e-47
```



- PSD averaged over many SFTs has massively reduced bin-to-bin variance
- Many different ways to do that averaging over SFTs:

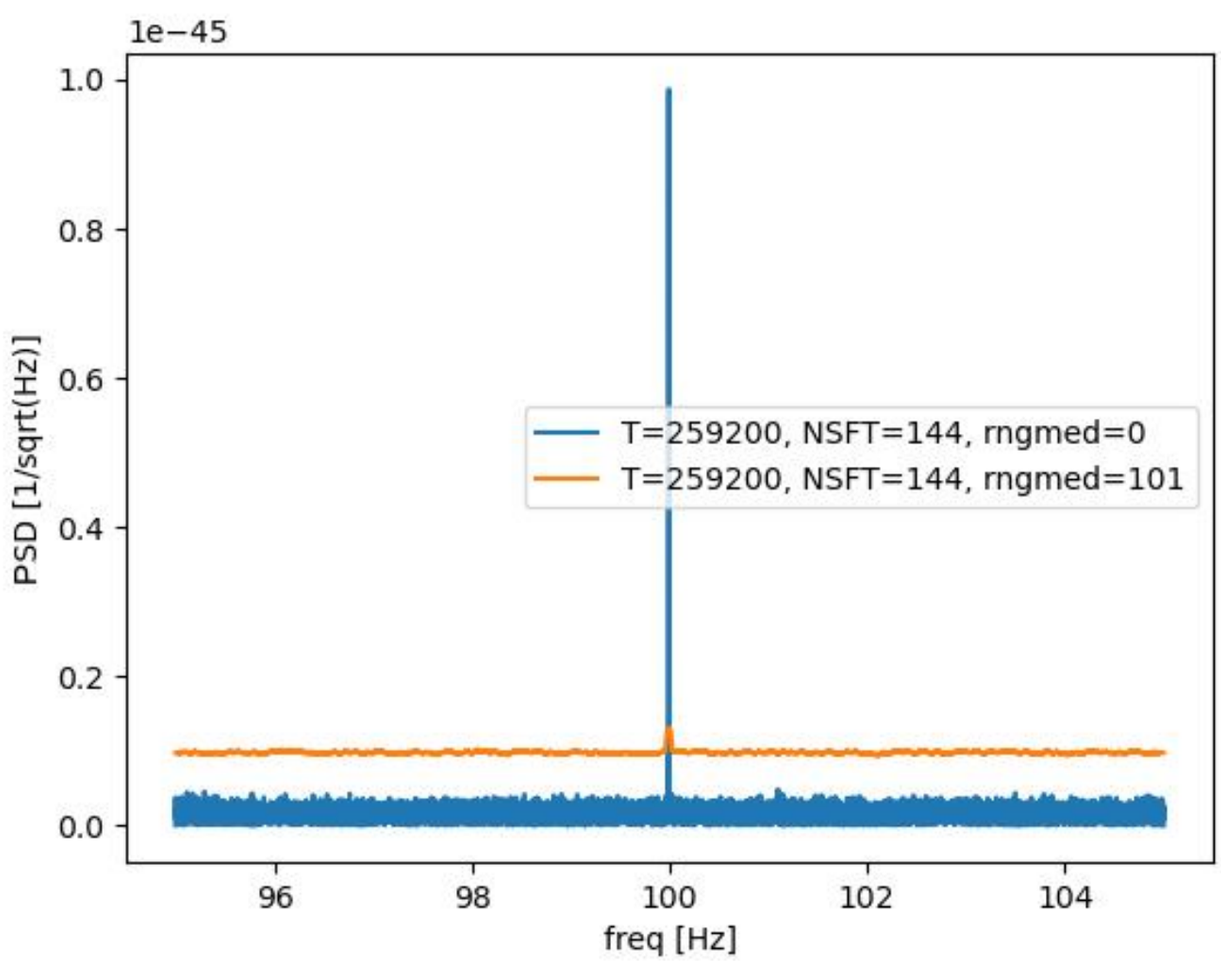
```
      -S, --PSDmthopSFTs=(arithsum|arithmean|arithmedian|harmsum|
harmmean|powerminus2sum|powerminus2mean|min|max|0|1|2|3|4|5|6|7|8)
[default: harmmean]
          For PSD, type of math. operation over SFTs, can be given by
string names (preferred) or legacy numbers:
          arithsum (0), arithmean (1), arithmedian (2), harmsum (3),
harmmean (4), powerminus2sum (5), powerminus2mean (6), min (7), max (8)
```

PSDs

- In addition to per-SFT averaging, by default our PSDs also have a *running median* applied over frequency bins:

```
-w, --blocksRngMed=<4-byte signed integer> [default: 101]
            Running Median window size
```

- This suppresses narrow instrumental lines, but if they're very strong, they will contaminate the background noise level estimate.



PSDs

- Like most CW *search* codes (e.g. `lalpulsar_ComputeFstatistic_v2` for fully-coherent *F*-stat searches, `lalpulsar_Weave` for semicoherent sarches), `lalpulsar_ComputePSD` also allows loading many SFT files together, *and from multiple detectors*.
- Same averaging operations as over SFTs from one detector can also be done over multiple detectors, but the best choice depends on your final application – generally, you want the PSD estimate to match in construction your search algorithm.

```
        -i, --inputData=<string> [required]
            Input SFT pattern
        --IFO=<string> [default: NULL]
            Detector filter
        -J, --nSFTmthopIFOs=(arithsum|arithmean|arithmedian|harmsum|
harmmean|powerminus2sum|powerminus2mean|min|max|0|1|2|3|4|5|6|7|8)
[default: max]
            For norm. SFT, type of math. op. over IFOs: see --
PSDmthopSFTs
```

real data processing

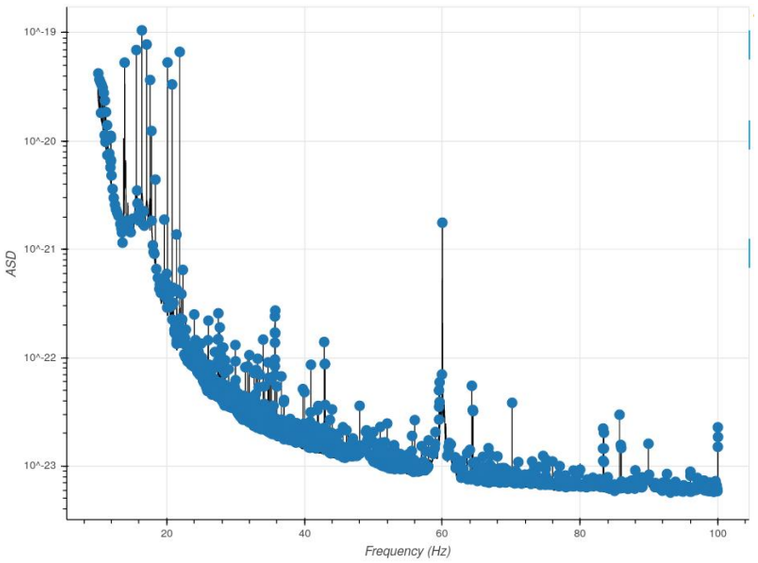
- Not going to lie: doing a CW search from scratch only using GWOSC and other public resources is *hard*. With a few notable exceptions, mostly only groups with previous within-the-LSC experience have pulled this off so far.
- Starting point: bulk strain download, e.g. for O3: https://gwosc.org/O3/O3a/ and https://gwosc.org/O3/O3b/
- This gives you time-domain data in .gwf frame format.
- Next, you need a *segment list* of “good” observing-mode times. These are published separately, e.g. for O3: https://dcc.ligo.org/T2300068/public
- You then need to make *cache files*, which have one line per input .gwf file like:

```
H H1_mfdv5 1238112018 1800 file://localhost//home/dkeitel/testdir/H-H1_mfdv5-1238112018-1800.gwf
```

- Then use `lalpulsar_MakeSFTs` or, for batch processing a whole run (and if you have a condor/DAG capable cluster) `lalpulsar_MakeSFTDAG`.
  You’ll need to figure out the right options for channel names, windowing, …
- LVK members can instead find pre-generated SFTs for all observing runs on the LDG (though not mirrored to all clusters), at least for Tsft=1800s.
- After bulk data release, these are OK to use in short-author papers, as long as the open data paper + the corresponding segment list are cited, and GWOSC as well as the CW group are acknowledged.

dealing with lines

- The LVK detchar group, observatory staff and a small team within the CW group spend significant effort mitigating spectral artifacts, and curating *line lists* that can be used by each CW search to either
  - do additional frequency-domain cleaning before the main analysis
  - veto outliers automatically (more on that later)
  - manually check final-stage surviving outliers
- These are maintained internally at https://git.ligo.org/CW/instrumental/aLIGO-lines-combs/ and publicly at
  - https://www.gw-openscience.org/o1speclines/
  - https://www.gw-openscience.org/o2speclines/
  - https://dcc.ligo.org/T2100200/public (O3 identified lines)
  - https://dcc.ligo.org/T2100201/public (O3 unidentified lines)



- Data formats can be a bit difficult to use, since many lines are parts of *combs* which then aren't listed individually in these files, but have to be expanded based on their base frequency, offset and number of harmonics.
- See Ansel Neunzert's tutorial!

now: PyFstat tutorials

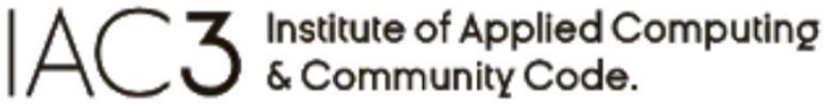
- If you don’t have a working environment with LALSuite and PyFstat yet, follow instructions at https://github.com/PyFstat/PyFstat/wiki/conda-environments or try Google Colab to run the tutorials online.
- Tutorials will be based on https://pyfstat.readthedocs.io/en/stable/examples.html
- PyFstat project home: https://github.com/PyFstat/PyFstat
- PyFstat reference: https://doi.org/10.21105/joss.03000
- Note: this is a relatively small project. It builds on top of LALSuite (thanks to Karl Wette’s SWIG bindings which allow python to call C libraries, Wette2020) but it is not as deeply tested as LALSuite itself, and only LVK-reviewed for a few specific applications (mainly MCMC candidate follow-up).

  → Only the second workshop where we run tutorials, and it’s quite possible that you run into some bugs in corner cases, or missing features.

  → Issue reports or pull requests via github are very welcome!

acknowledgments

David Keitel is supported by the Spanish Ministerio de Ciencia, Innovación y Universidades (ref. BEAGAL 18/00148) and cofinanced by the Universitat de les Illes Balears. Rodrigo Tenorio is supported by the Spanish Ministerio de Universidades (ref. FPU 18/00694). This work was supported by the Universitat de les Illes Balears (UIB); the Spanish Ministry of Science and Innovation (MCIN) and the Spanish Agencia Estatal de Investigación (AEI) grants PID2019-106416GB-I00/MCIN/AEI/10.13039/501100011033, RED2022-134204-E, RED2022-134411-T; the MCIN with funding from the European Union NextGenerationEU (PRTR-C17.I1); the FEDER Operational Program 2021-2027 of the Balearic Islands; the Comunitat Autònoma de les Illes Balears through the Direcció General de Política Universitaria i Recerca with funds from the Tourist Stay Tax Law ITS 2017-006 (PRD2018/23, PDR2020/11); the Conselleria de Fons Europeus, Universitat i Cultura del Govern de les Illes Balears; and EU COST Action CA18108. This material is based upon work supported by NSF's LIGO Laboratory which is a major facility fully funded by the National Science Foundation.



Unión Europea

Fondo Europeo
de Desarrollo Regional
"Una manera de hacer Europa"