

XIII ET Symposium

Cagliari, 8-12 May 2023

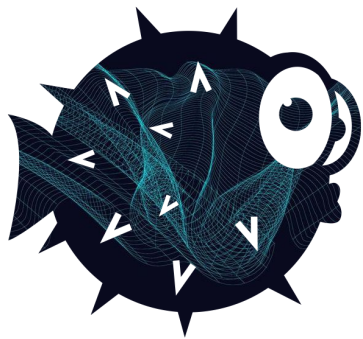


GWFish

A Fisher Matrix Analysis Software

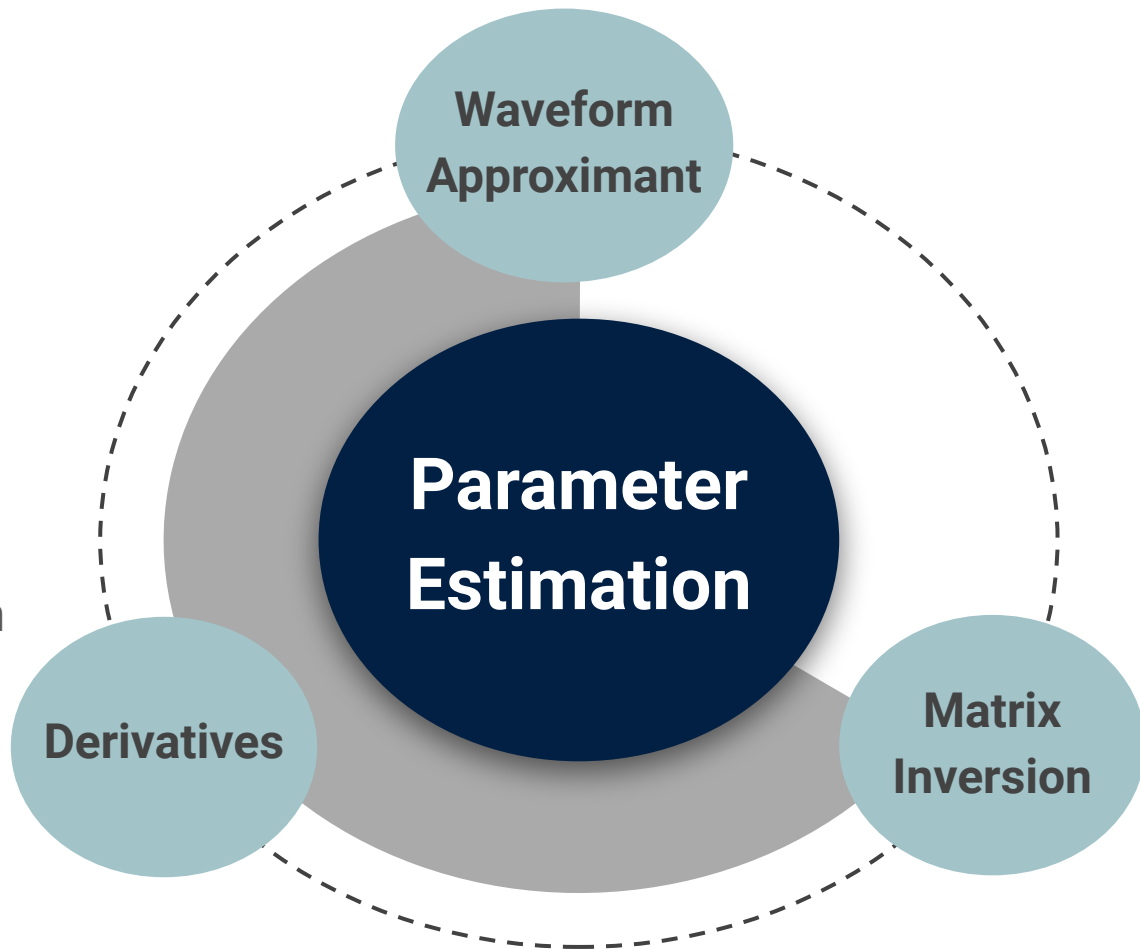


Ulyana Dupletsa
Jacopo Tissino
Francesco Iacovelli

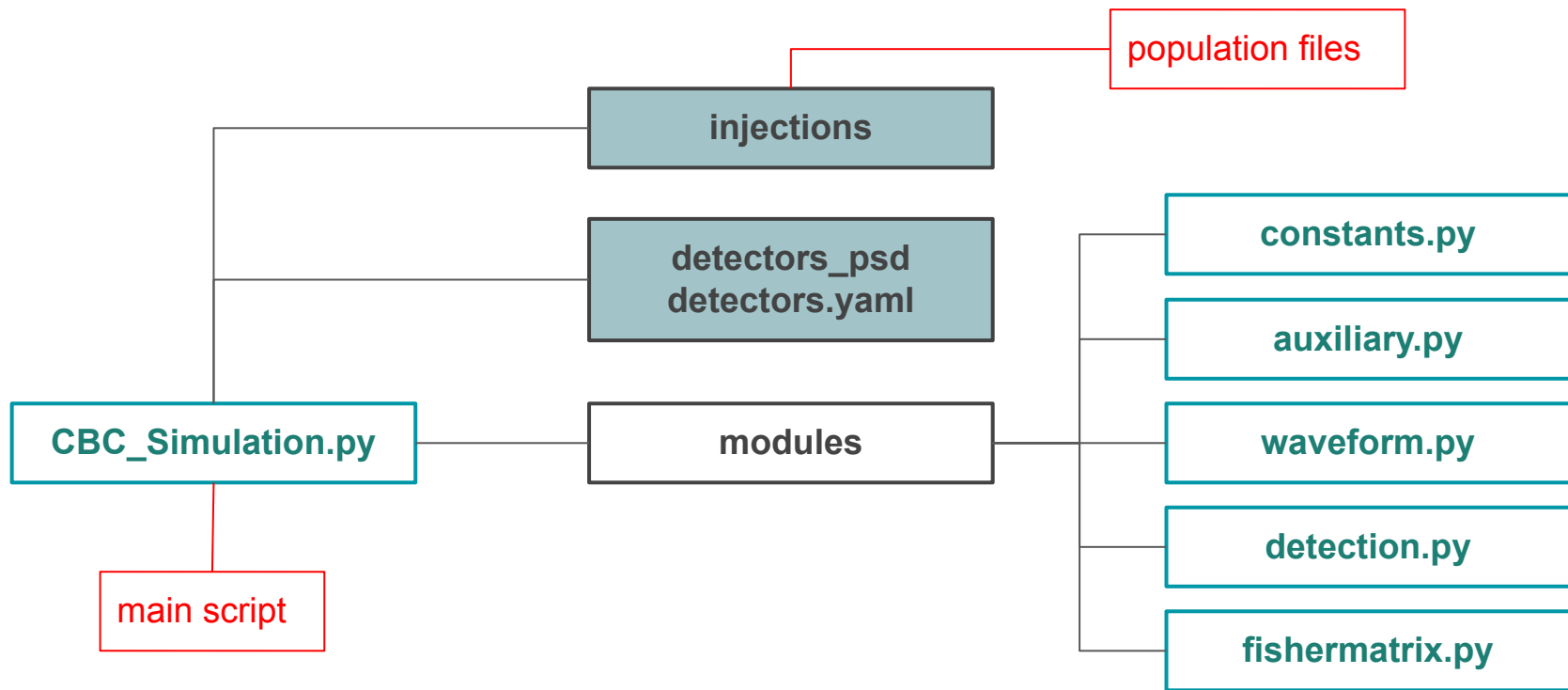


GWFISH

- Written in Python
- Implementation relies on three main ingredients:

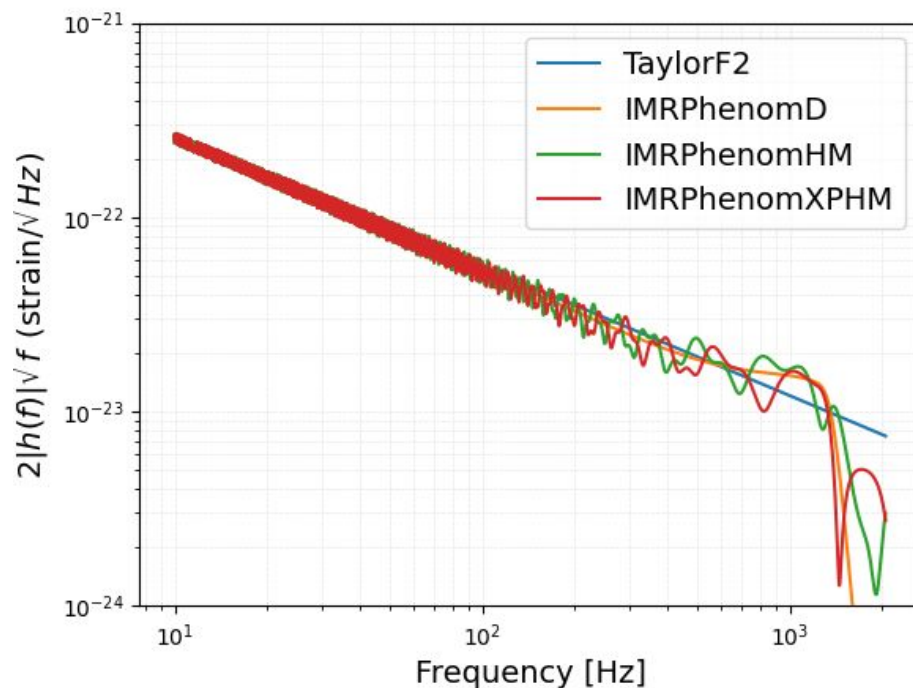


Basic Structure



GWFish Waveforms: link to LALSimulation

- All waveforms approximants from LALSimulation:
 - TaylorF2
 - IMRPhenomD
 - IMRPhenomXPHM
 - IMRPhenomNR_Tidalv2
 - ...
- Independent implementation of TaylorF2 and IMRPhenomD waveform approximants



Derivative Implementation in GWFish

- Numerical differentiation suffers from limitations of numerical precision
- Analytic derivatives with respect to:
 - Waveform phase at merger
 - Merger time
 - Luminosity distance
- Derivatives with respect to all the other parameters are calculated numerically:

```
pv_set1[p] = pv - dp / 2.  
pv_set2[p] = pv + dp / 2.
```

```
signal1 = det.projection(pv_set1, detector, wave, t_of_f)  
signal2 = det.projection(pv_set2, detector, wave, t_of_f)
```

```
derivative = (signal2 - signal1) / dp
```

Matrix Inversion in GWFish: SVD

Normalize matrix before inversion

```
dm = np.sqrt(np.diag(matrix))  
normalizer = np.outer(dm, dm)  
matrix_norm = matrix / normalizer
```

Discard singular eigenvalues smaller than $1e-10$

```
[U, S, Vh] = np.linalg.svd(matrix_norm)  
  
kVal = sum(S > thresh)  
matrix_inverse_norm = U[:, 0:kVal] @ np.diag(1. / S[0:kVal]) @ Vh[0:kVal, :]  
  
return matrix_inverse_norm / normalizer
```

Undo the normalization

Setting up a Run with GWFish

```
parser.add_argument(  
    '--pop_file', type=str, default='./injections/CBC_pop.hdf5')  
parser.add_argument(  
    '--detectors', type=str, default=['ET', 'CE'], nargs='+')  
parser.add_argument(  
    '--networks', default='[[0,1]]')  
parser.add_argument(  
    '--config', type=str, default='GWFish/detectors.yaml')
```

← population file

← detectors

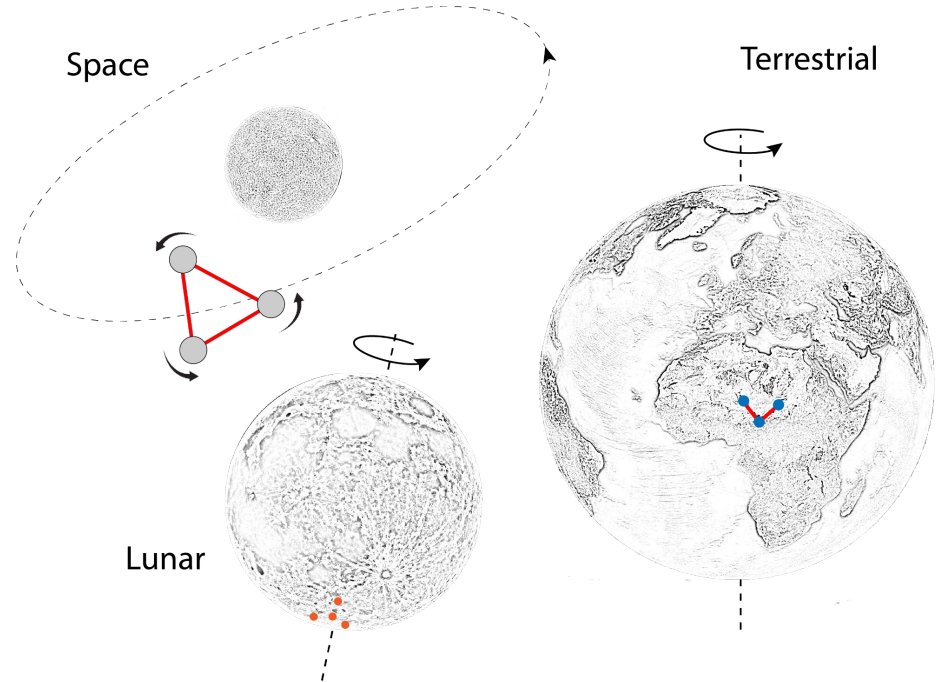
← detector networks

← detector config file

```
threshold_SNR = np.array([0., 8.])  
calculate_errors = True  
duty_cycle = False  
  
fisher_parameters = ['theta_jn', 'luminosity_distance', 'mass_1', 'mass_2']  
  
waveform_model = 'lalsim_IMRPhenomXPHM'
```

Detector Networks available in GWFish

- Potential upgrades of the current infrastructures: **Virgo**, **LIGO** (Hanford and Livingston), **LIGO India** and **KAGRA**
- The proposed **Einstein Telescope** and **Cosmic Explorer** (both second half of 2030s)
- The approved space-borne detector **LISA**, expected to begin observations in the second half of 2030s
- New detector concept on the lunar surface: **LGWA**



Detector Settings

```
ET:
  lat:      (40 + 31. / 60) * np.pi / 180.
  lon:      (9 + 25. / 60) * np.pi / 180.
  opening_angle: np.pi / 3.
  azimuth:   70.5674 * np.pi / 180.
  psd_data:  ET_psd.txt
  duty_factor: 0.85
  detector_class: earthDelta
  plorange:  3, 1000, 1e-25, 1e-20
  fmin:      1.
  fmax:      2048.
  spacing:   geometric
  df:        1./16.
  npoints:   5000
```

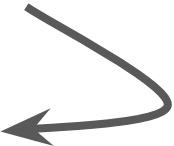
All the detectors' settings
are listed in the
detectors.yaml file

Example of variables needed
to set up a detector

These define the
frequency vector (from
fmin to fmax) and one can
choose either a **linear** or
a **geometric spacing**

GWFish Output Files

1. **Signals.txt**: injection parameters + SNR of all the chosen detector networks
2. **Errors.txt**: parameters **1 sigma** errors + 1 sigma **sky localization** (if RA and DEC are among the chosen Fisher parameters)

$$\Delta\Omega = \pi |\cos(DEC)| \sqrt{\sigma_{RA}^2 \sigma_{DEC}^2 - \sigma_{RA,DEC}^2}$$


3. Fishers.npy: list of Fisher matrices for the events above the SNR threshold
4. Inv_Fishers.npy: list of covariance matrices for the events above the SNR threshold
5. Sing_Values.npy: list of singular eigenvalues for each of the Fisher matrix in Fishers.npy file

Thanks!