Proposed GWHEN analysis

Triggered search

GWHEN analysis joint LVK-KM3Net - Report

People involved:

~~Pure~~ LVK analyses | Search for a GW counterpart of KM3Net triggers:
List of triggers needed to perform the analysis (lara). Tools X-pipeline and PyGRB

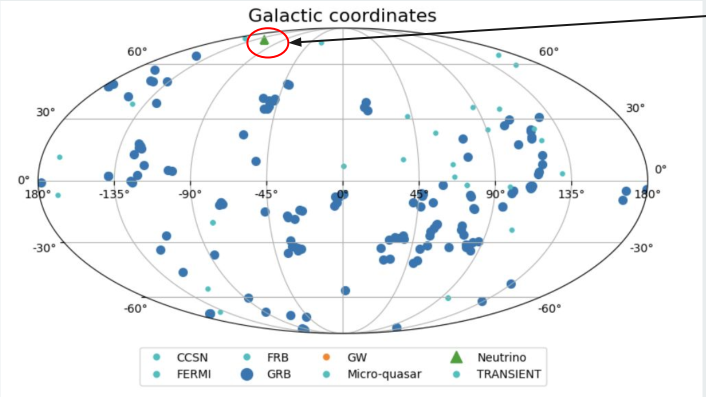
Not "pure" LVK: Mathieu from UCLouvain is also involved (data and discussion from Marion's work)
Marion and Matthias

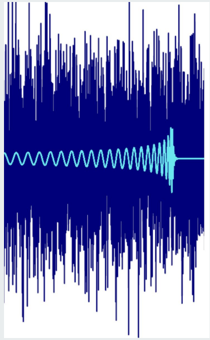
Types of analysis: Triggered search - based on sky localization and time of an HEN trigger
Pipelines: X-pipeline and PyGRB
Offline analysis

GWHEN analysis joint LVK-KM3Net - Report

Triggered analysis both modelled and unmodelled

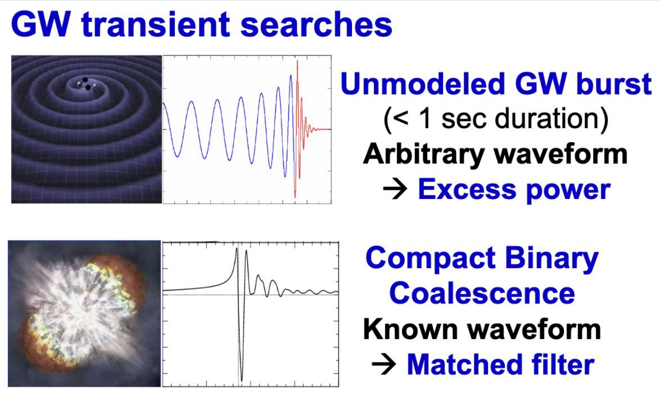
In short:
We take to sky position and time of the trigger and look if there was a GW in coincidence





Know the sky position, time of event and
Searches for GW transient signals

known and unknown waveforms



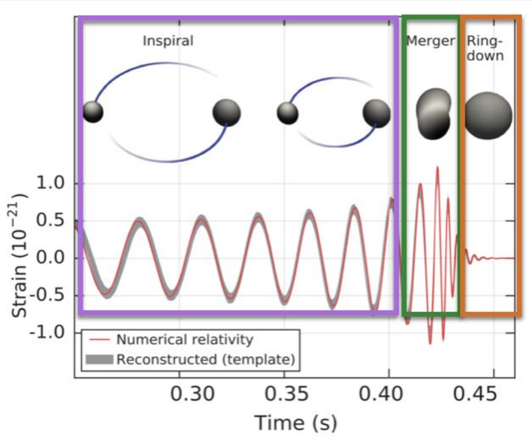
Performed with
X-pipeline

Performed
with PyGRB

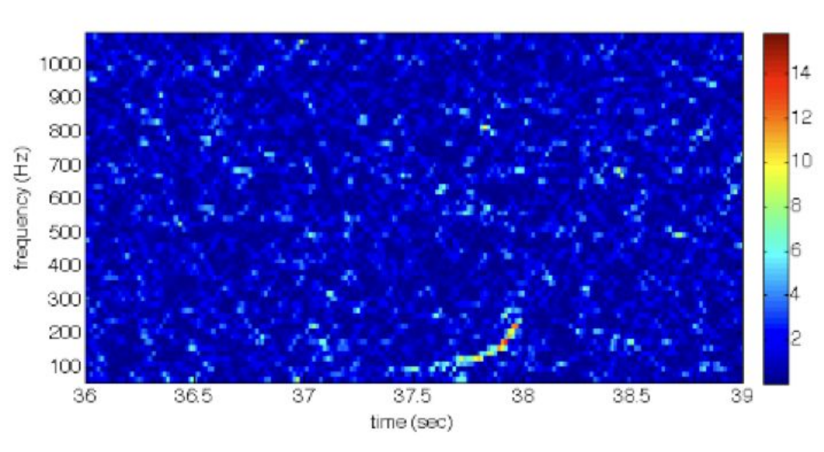
GWHEN analysis joint LVK-KM3Net - Report

Triggered analysis both modelled and unmodelled

PyGRB models waveforms based on:



X-pipeline looks for excess of power on the detectors background:



Goal:

- perform a targeted search using the times of the HEN and their sky localizations to search for possible GW associations.
- use different emission models to put a lower bound on the distances of the HEN progenitors.
- compare the results are with the previously predicted detection rate for GWHEN

GWHEN analysis joint LVK-KM3Net - Report

Triggered analysis with unmodelled tool

Status: phase of designing the analysis and making some decisions, such as:

- Pipelines to be used as first: maybe just X-pipeline for KM3Net O3 data if used…
- Waveforms to be simulated: long GRBs, inspirals. *Short GRBs burst from f-modes, sine gaussians?* *MOST LIKELY COINCIDENCE OF GW AND HEN FROM LONG GRBs*
- Time window: -500;500 for X-pipeline = in line with RAVEN and previous GWHEN searches

DONE!!

Next up:

- Ready to run with X-pipeline
- Info needed:
  - KM3Net triggers catalog of **well-localized** neutrinos only during O3 (data shared so far has a huge number of trigger which is can not be used as the computational time required for X-pipeline is big)
  - Provided the triggers with real time no shuffled time
  - Goal end up with a list of several triggers (2 or 3 per month of data) - need to check the person power to run X-pipeline
  - *If such criterias are not enough, other criteria will be needed - **IDEAS?***

From the data Marion has, it is possible to compute the area in deg2 of the error box of the triggers (Marion is doing data already)

Document related to the analysis: https://docs.google.com/document/d/1UGTWUUCFJ-L-5onCfcZLkJYdP3pfLnutt0pHeq6gx-g/edit

GWHEN analysis joint LVK-KM3Net - Report

Triggered analysis with unmodelled tool .INI file (design of the analysis) ready and tested FOR UNMODELLED ANALYSIS (WITH X-PIPELINE)

```
; ---- Draft X-Pipeline parameter file for GWHEN O3b offline searches - based on grb o3b .ini file
;      See https://trac.ligo.caltech.edu/xpipeline/wiki/Documentation/Searches/grb/grb.iniDoc
;      Note that some of the files pointed to here are actually for O3a and need to be updated.


[tags]
; ---- Optional tag for this file/run. This is dumped to stdout by grb.py but otherwise is not used.
version = $Id$


[background]
; ---- Information on data to be used for background estimation.
; ---- Duration of background period in seconds. This length of time, by default centered on
;      the trigger, is used for background event generation.
backgroundPeriod = 10800
; ---- This is needed by the online search when using asymmetric background estimation.
;      If doAsymmetricBackground = 0 then the background period is symmetric about the trigger time.
;      Otherwise backgroundAsymmetryFactor is the fraction of the background period before the trigger ti
;      Specifically, the background period is [start_time,end_time] where
;        start_time = int(trigger_time - background_period*backgroundAsymmetryFactor)
;          end_time = int(trigger_time + background_period*(1.0 - backgroundAsymmetryFactor))
doAsymmetricBackground = 0
backgroundAsymmetryFactor = 0.9
; ---- Full path of files listing time lags to be used for background estimation.
;      Specify one lag file for each network type. (Note that lag files
;      are ignored for single-detector networks.)
;      The format is one set of time lags per row, with each column being the time shift applied
;      to the corresponding detector in the [input] detectorList below. With variable names of
;      the form "lags_2det1site" etc., grb.py will automatically select the right file based on
;      what detectors are being used.
;      TIP: It's good practice to point to a file in a repository if running an analysis to be reviewed.
;      All coincidence segments for all listed time lags will be used to generate
;      background. The actual number of background trials you get isn't determined
;      advance, as it depends on the duty cycle of each detector, and also on the value of
;      circtimeslidestep in the [parameters] section.
; ---- Use the same lag files as in O3a:
lags_2det2site = /home/xpipeline/xpipeline/branches/scripts/input/O3b/lags_2det2site_20.txt
lags_3det3site = /home/xpipeline/xpipeline/branches/scripts/input/O3b/lags_3det3site_20.txt


[parameters]
; ---- xdetection parameters. See
;      https://trac.ligo.caltech.edu/xpipeline/wiki/Documentation/xdetection
;      Particularly important parameters that are not obvious are:
;        onSourceBeginOffset,onSourceEndOffset - these define the on source region to be analyzed as
;          [grbTriggerTime+onSourceBeginOffset,grbTriggerTime+onSourceEndOffset], where grbTriggerTime
;          is the GPS time supplied by the -g option of grb.py
;        circtimeslidestep - this is the step size of circular time shifts applied within each
;        background job. Smaller values give more circular time shifts and more background trials.
analysisTimes = 2.0,1.0,0.5,0.25,0.125,0.0625,0.03125,0.015625,0.0078125
blockTime = 256
onSourceBeginOffset = -500
onSourceEndOffset = 500
likelihoodType_1det1site = loghbayesian,energyitf1,skypositiontheta,skypositionphi
likelihoodType_2det2site = loghbayesiancirc,standard,circenergy,circinc,circnullenergy,circnullinc,powerla
likelihoodType_3det3site = loghbayesiancirc,standard,circenergy,circinc,circnullenergy,circnullinc,nullene
minimumFrequency = 20
maximumFrequency = 500
```

```
[injection]
; ---- This section specifies properties common to all injection waveform sets,
;      such as amplitude scales and whether to account for calibration uncertainties.
; ---- Amplitude scales applied to all injections in the [waveforms] section.
injectionScales = 0.0100,0.0147,0.0215,0.0316,0.0464,0.0681,0.1000,0.1468,0.2154,0.3162,0.4642,0.6813,1.00
; ---- This parameter controls the number or spacing of injections.
;      Positive value: time interval between injections. The number of injections at each
;        injection scale is approximately blockTime/injectionInterval.
;      Negative value: on-the-fly injections are spaced so that exactly
;        abs(injectionInterval) injections are performed at each injection scale.
injectionInterval = -1200
; ---- Apply estimated calibration uncertainties to injections. This should
;      always be 1 for production analyses.
miscalibrateInjections = 1


[waveforms]
; ---- Sets of on-the-fly injections to be performed. Fixed amplitudes.
;      Format: set_name = waveform_type!waveform_parameters
;          or: set_name = waveform_type1!waveform_parameters1,waveform_type2!waveform_parameters2,...
;        set_name - an arbitrary string of letters and numbers only (no underscores,
;          punctuation marks, etc.).  Make sure that each name is unique and not a sub-string of
;          another name. This is because some book-keeping codes use "ls" to find files associated
;          with a given injection set. So, for example, having sets dfm1, dfm2, dfm3 is fine, but
;          sets dfm, dfm2, dfm3 will cause failures because a command like "ls dfm*" for the first
;          injection set will also return files associated with dfm2, dfm3. And be careful if you
;          have many sets: dfm1, ..., dfm10!
;        waveform_type - can be any of the types recognized by xmakewaveform; type
;          "help xmakewaveform" in matlab.
;        waveform_parameters - To understand the parameters of a given type, e.g. 'chirplet',
;          do "xmakewaveform('chirplet')" in matlab. A parameter of the form 'number;number;word'
;          (e.g. -1;1;linear) will cause the value of the parameter to be generated randomly for
;          each injection. For allowed syntax see the 'assignparameter' helper function of
;          https://trac.ligo.caltech.edu/xpipeline/browser/xpipeline/trunk/matlab/share/xmakegwbinjectionf
;      If there is more than one waveform in the comma-separated list, then for each injection one
;      waveform is chosen randomly from the list.
;      Note that some waveforms are loaded from catalogs; in these cases you
;      must use the -c option with grb.py to specify a location. All valid
;      catalogs are stored in branches/waveforms/ .
sgc70Q9  = chirplet!1.0e-22~0.0143~70~0~0~-1;-0.996;linear
sgc100Q9 = chirplet!1.0e-22~0.01~100~0~0~0.996;1;linear
sgc150Q9 = chirplet!1.0e-22~0.006667~150~0~0~-1;-0.996;linear
sgc300Q9 = chirplet!1.0e-22~0.00333~300~0~0~0.996;1;linear
bns = inspiral!1.4;0.2;1;3;1.4;0.2;1;3;2;6;mass~0.866;1;linear~10
nsbh = inspiral!1.4;0.2;1;3;10;6;2;25;3;25;mass~-1;-0.866;linear~20
adi-a  = adi-a!10~0.996;1;linear
adi-b  = adi-b!20~-1;-0.996;linear
adi-c  = adi-c!10~0.996;1;linear
adi-d  = adi-d!10~-1;-0.996;linear
adi-e  = adi-e!10~0.996;1;linear
```