Document NZGMDB Process

Aim is to have a record of the entire NZGMDB process so it is clear on what exactly is being done.
CURRENTLY IN DRAFTING PHASE SO WILL BE RANDOM QUICK NOTES

Geonet Mseed Creation:

1: Downloads the recent earthquake data from geonet within the given date range
2: Only uses strong motion channel codes HN and BN for gettin the FDSN Client data
3: Loads the mw_rrup txt file and uses an interpolation function "cubic"

When fetching the event line for the source table
Gets the preferred_origin and magnitude from the event catalouge Geonet client

If the prefer_mag_type is m then gets all the event cat magnitudes and finds the max for mb ml and mlv
if there is one for mb and less than 3 station counts then find max station count for ml and mlv and compare station counts to determine the loc_mag, otherwise make it mb
If there is no mb then find max station count out of ml and mlv
If there if none of mb, ml or mlv then try m

Otherwise if it isnt m for the prefer_mag_type then just set the magnitude uncertainty values to the preferred magnitude values

When grabbing the station magnitude lines it gets the stations within a given radius
This is based on the Mw_rrups.txt file which cuts off at each end instead of extrapolation.
Then the inventory is selected using the max_radius from the event lat lon values.

The inventory is then looped over and the waveform is extracted from the client.
The range of the waveform is determined by the p_arrival from the following code

```
TauPyModel(model="iasp91")p_arrivals = model.get_travel_times(
source_depth_in_km=preferred_origin.depth / 1000,
distance_in_degree=deg,
phase_list=["ttp"],
)
s_arrivals = model.get_travel_times(
source_depth_in_km=preferred_origin.depth / 1000,
distance_in_degree=deg,
phase_list=["tts"],
)ptime_est = (
preferred_origin.time + p_arrivals[0].time
) # Estimated earliest P arrival time from taup
stime_est = preferred_origin.time + s_arrivals[0].time
```

The start time is ptime_est - the in_time_difference which is set to 15 in the config
The end time is based on the following

```
    end_time = stime_est + (
        min_time_difference
        if stime_est + ds * ds_multiplier - ptime_est < min_time_difference
        else ds * ds_multiplier
    )
```

Where ds is the output from the AS_16 model from openquake (previously was using a custom model that had incorrect coefficients)
ds multiplier is currently 10 but needs investigation as there may be a better method to ensure waveforms arent cut off.

Only channel codes that are selected are HN? and BN?
All locations are also taken for mseed file creation
If there is no data or the file is too small then the mseed file won't be written, otherwise it attempts 3 times with network issues, if that fails 3 times then just skips

For the station magnitude line the station magnitude is obtained from the Z channel where the first 2 channel codes match if possible otherwise is set to None and the type is set to the pref_mag_type

Multiprocessing per event_id
Outputs the earthquake_source_table and station_magnitude_table

Note: check for response adding the same for all channels

Tectonic Domain Addition:

Takes the earthquake source table as an input and a bunch of shape files with CMT and NZSMDB data
Updates each entry in the event_df if the same evid is found in the CMT data "GeoNet_CMT_solutions.csv". Updates the ["mag", "lat", "lon", "depth"], notifies in the mag-method to be CMT
Then merges the NZSMDB where the evid matches
Creates 3 region areas based on if the lat lon points are offshore to the fault, on the fault or downdip to the fault. With the Kermadec and Hikurangi and Puysegur datasets. d_s and d_d values of 10, 47, 11, 30 respectively
Regions are based on this

```
df_a = df[(df.depth < d_s)]
df_b = df[(df.depth >= d_s) & (df.depth <= d_d)]
df_c = df[(df.depth > d_d)]
```

Then the regions are used with a horizontal and vertical threshold from each lat lon to each point on the region and depth to determine the tect class

```
    Region A (vertical prism offshore of seismogenic zone of fault plane):
    depth<60km: 'Outer rise'
    depth>=60km: 'Slab'

    Region B (vertical prism containing seismogenic zone of fault plane):
    depth<min(slab surface, 20km): 'Crustal'
    min(slab surface, 20km)>depth>60km: 'Interface'
    depth>60km: 'Slab'

    Region C (vertical prism downdip of the seismogenic zone of the fault plane):
    depth<30km: 'Crustal'
    30km<depth<slab surface: 'Undetermined'
    depth>slab surface: 'Slab'

    Elsewhere (Farfield):
    depth<30km: 'Crustal'
    depth>30km: 'Undetermined'
```

If the NZSMDB then has a tect_class defined for the given evid then it is replaced and the tect_method is set to be the NZSMDB.
Then the CMT is merged is there is data found from the CMT file on each evid
Then the domain is found based on the fact of if the lon lat is within the region from the layers in a shape, each has a number and a type associated with it.
Uses the NZTM coordinate system.

Phase_arrival:

p wave picker and a gen phase arrival for creating the table needed for output csv

Need to add in the phase arrival information from geonet
Not sure about t_res - This can be just there for applicable data and removed for custom p picker
If there is a clash with the Geonet P wave and the custom picker then choose the custom picker
arid can be removed and avid can be the first column
channel needs to be more generic and just first 2 characters

mseed file PosixPath('/home/joel/local/gmdb/testing_folder/waveforms/2021/2021p001797/mseed/2021p001797_PAWZ_EH.mseed') has 2 P wave picks but are both exactly the same

SNR Calculation:

Takes in an input of a phase arrival table and mseed files to write SNR files in the snr_fas directory
The directory is layered like waveforms, year / event_id / snr_fas.csv files
A skipped files csv is also generated as a flatfile to notify which files were skipped from the process

A common frequency vector is used which is defined as
0.01318257 100 (with 389 numbers)

Applies initial processing on the record such as

- Demean and detrend
- Taper both ends by 5%
- Add zero padding of time 5s
- Remove sensitivity based on the Inventory (If failed to find will skip record)
- Rotate the components to NEZ (can still be XYZ?)
- Divide the data by gravity

Grabs the tp from the phase arrival table. (Index in the waveform where the p-wave arrives not just datetime)
Will skip the record if there is no entry in the table for the p-wave
Or if the tp value does not fit in the bounds of the given waveform

If all these steps pass then SNR / FAS is calculated as below
First separates into signal and noise based on the tp value
Checks that the noise duration is not under 1s (If so then the record is skipped)
Then applies a hanning taper to the start and end of the waveform 5% to both the signal and noise
the fa_spectrum is then calculated for signal and noise
The absolute value is then taken of this output
If smoothing is enabled which is by default True (Might need to be added as an option for the full workflow?) Then applies ko_matrix smoothing b = 40
Interpolation is then done on the FAS to the common frequency vector
We then set values to NAN for the frequencies that are outside the bounds of the sample rate / 2
Then we calculate SNR (inter_signal / inter_noise is the interpolated signal / noise)

```
snr = (inter_signal / np.sqrt(signal_duration)) / (
        inter_noise / np.sqrt(noise_duration)
    )
```

The output is the snr_fas files in the given event directory as well as a single metadata csv in the flatfiles directory with the skipped records and reasons why they were skipped.

TODO

Change SNR output names, remove last channel and add loc

Random SNR write issue, leaking into extra columns? Might be a once off strange error, maybe a check to redo if persist

2 tapers for processing?? Do we need to specify no taper or zero padding in initial processing?

Fmax calculation:

Still to be intergrated into the workflow
Currently have a converted matlab script into python that is hacked together, but needs work on intergration.

Extra steps:

- Fmax output column rename to stem of the mseed file (instead of ev_sta_chan, have a record_id column)
- Add a skipped records csv and list reasons why the snr file was skipped for fmax calc
- Add magic numbers to the config

I'd say the steps for this also need to be investigated and approved by the wider team (later work to be done)

GMC:

Blackbox, just requires mseed files
Outputs a gmc_predictions.csv file that goes into the flatfiles directory

TODO

Make work with the full workflow (Upgrade to tensorflow2)
GMC station value is randomly the channel?
Ensure proper pre-processing is done on waveforms for determining fmin

Process records:

Takes a fmax file and the gmc predictions for fmin and fmax and requires mseed files

A quick check is done that there is all 3 components (This is some bug that needs to be fixed in geonet.py)

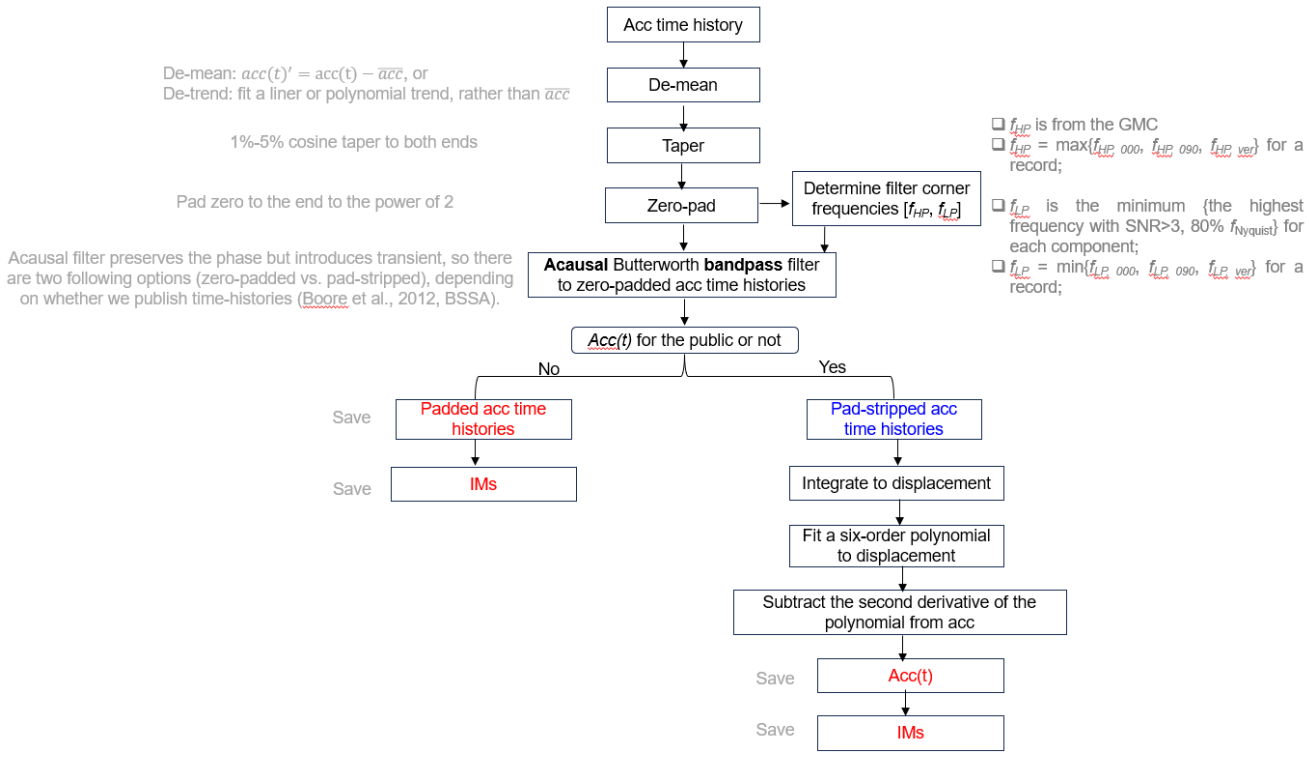
Initial pre_processing (Same as whats written in SNR) is then performed and processing is skipped if the Inventory failed to fetch or failed to remove sensitivity

fmax and fmin values are then extracted from the gmc and fmax dataframes
For fmin the max fmin_mean value is taken (max across all 3 components)
fmax is the min of all 3 components

If these values are not found then for the fmin, lowcut is set to 0.05 and if fmax is missing then the highcut is set to 1 / (2.5 * dt)
A simple check is performed that the lowcut is not higher than the highcut, if so the record is skipped

Then the 000, 090 and ver components are selected from the initial processed mseed (rotated)
Tries to grab NE if not then tries XY and if not then skips record

Then applies the following for processing



- butter bandpass filter - order of 4 fs = 1 / dt
- Remove zero padding
- Calculate velocity
  `vel_000 = integrate.cumtrapz(y=acc_bb_000, dx=dt, initial=0.0) * g / 10.0`
- Calculate displacement
  `disp_000 = integrate.cumtrapz(y=vel_000, dx=dt, initial=0.0)`
- Fit a six-order polynomial

  `coeff_000 = np.polyfit(np.arange(len(disp_000)), disp_000, poly_order)`
- Find the 2nd derivative of the coefficients
  `coeff_000_2nd = np.polyder(coeff_000, 2)`
- generate polynomial from the coefficients
  `poly_000 = np.polyval(coeff_000_2nd, np.arange(len(acc_bb_000)))`
- Subtract the polynomial from the original acc series

After all that saves as individual component files named the same as the mseed but with the extentions of .000 .090 and .ver in the waveforms directory but under "processed"

IM Calculation:

```
psa_periods = np.asarray([0.01, 0.02, 0.03, 0.04, 0.05, 0.075, 0.1, 0.12, 0.15, 0.17, 0.2, 0.25, 0.3, 0.4, 0.5, 0.6
, 0.7, 0.75, 0.8, 0.9, 1.0, 1.25, 1.5, 2.0, 2.5, 3.0, 4.0, 5.0, 6.0, 7.5, 10.0])
Custom hacky script
Might need a rework of compute_,measure_multiprocess or just use the single compute seperatly for NZGMDB due to
different file paths?
```

Maybe remove the gm_all? - Maybe just rename or combine by default
If we instead of station and component for the multi process added record_id like the mseed then merging is easier

Add in the year to output directory steps for consistancy
Might need to rename outputs due to same stations can have different locations and channels for outputs

Merge Faltfiles:

Original steps:

- gm_im_merge
- concat_ims
- calc_distance_gmdb
- merge_all