QUEEn's Fourier Filtering (QUEEFF) code
Quickstart guide

David B. Jess[†] & Samuel D. T. Grant

The process of Fourier filtering time series has been a key analysis tool in both the detection of wave signatures within solar data, including the removal of unwanted noise sources within the data, for decades. Traditionally, the use of Fourier methods focused solely on the (temporal) frequency domain of the data, preserving frequencies known to contain significant wave power, whilst suppressing others with minimal Fourier power and/or associated with noise. However, a powerful yet previously neglected technique is to consider the spatial wavenumbers of an image (k_x & k_y, where $k = 2\pi/\lambda$, with λ the wavelength of the oscillation component). With this in mind, the typical length scale of an object in a sequence of images, such as a sunspot, can be selected, allowing the removal of any sources of oscillatory power at smaller and/or larger spatial scales. The application of combined spatial *and* temporal (i.e., '3D') Fourier filtering is documented in Jess et al. (2017), and the code derived for that study allows:

1. Spatial and temporal (3D) Fourier filtering, either in combination or individually, of a three-dimensional cube of images, with the output of the code being a filtered cube of the same dimensions as the input;
2. A visual mechanism for selecting the boundaries of the applied filter through inspection of a data-specific $k - \omega$ diagram (e.g., if the user is initially unsure of their required inputs);
3. Multiple graphical outputs throughout the Fourier filtering to visualise the filtering processes at work; and
4. Optional application of Gaussian filters to smooth the filter edge transitions and mitigate the degree of aliasing found in the filtered image sequence.

Below is a step-by-step guide to utilising the Fourier filtering algorithms employed by Jess et al. (2017). Text that is written in red font are inputs that you will need to enter, either at the command prompt (commencing with a $ symbol) or within an SSWIDL session (indicated by the IDL> start). Provided alongside the code are sample SDO/AIA image cubes if you wish to test out the functionality of the code before applying to your own datasets. Note that the time taken for the code to run is dependent on the dimensions of the input time series. Therefore, it is recommended that a small cube (such as the reference AIA image sequences) is used to become familiarised with the practicalities of Fourier filtering before applying to a larger dataset.

NOTE: If you employ the QUEEFF code you **MUST reference the first-use publication that it was designed for (Jess et al., 2017). References are in the bibliography at the end of this guide.**

Step 01:

Download the package and move to a temporary directory:

```
$ mv QUB_QUEEFF_package.tar.gz  /Users/QUB/Downloads/temp/
```

Step 02:

Extract the package to the temporary directory:

```
$ cd  /Users/QUB/Downloads/temp/
$ gunzip  QUB_QUEEFF_package.tar.gz
$ tar  -xvf  QUB_QUEEFF_package.tar
```

Once uncompressed, the package should contain two directories:

[†]Astrophysics Research Centre, School of Mathematics and Physics, Queen's University Belfast, Belfast, Northern Ireland, BT7 1NN, U.K. — d.jess@qub.ac.uk

QUEEFF_IDL	
QUB_QUEEFF.pro	The main filtering code
QUEEFF_dependencies.bat	An IDL batch file to compile all necessary constituents and alert any problems
QUEEFF_dependencies	A directory with non-standard IDL codes necessary for the filtering to work

Reference_Data	
Halpha_sunspot_COMPLETE.fits	HαRDcam Hα FITS file containing a sample $[512, 512, 2528]$ array with a default $0\farcs138$ pixel scale and a 1.78 s cadence
Halpha_sunspot_temporal_degrade.sav	HαRDcam Hα IDL save file containing a sample $[512, 512, 252]$ array with a default $0\farcs138$ pixel scale and a 17.8 s cadence
Halpha_sunspot_spatial_degrade.sav	HαRDcam Hα IDL save file containing a sample $[128, 128, 2528]$ array with a default $0\farcs552$ pixel scale and a 1.78 s cadence
Halpha_sunspot_umbra_apodized.sav	HαRDcam Hα IDL save file containing a sample $[512, 512, 252]$ array with a Hamming filter applied around the sunspot umbra, and a default $0\farcs138$ pixel scale and a 17.8 s cadence
aia1700.sav	SDO/AIA 1700 Å IDL save file containing a sample $[350, 350, 373]$ array with a default $0\farcs6$ pixel scale and a 24 s cadence
aia171.sav	SDO/AIA 171 Å IDL save file containing a sample $[350, 350, 750]$ array with a default $0\farcs6$ pixel scale and a 12 s cadence

The data products supplied here are Hydrogen-Alpha Rapid Dynamics camera (HαRDcam; Jess et al., 2012) and SDO/AIA observations acquired on 2011 December 10, and have been employed in multiple previous publications (e.g., Krishna Prasad et al., 2015; Jess et al., 2016; Krishna Prasad et al., 2017; Jess et al., 2019). Specifically, six IDL compatible files are provided:

- `Halpha_sunspot_COMPLETE.fits`
- `Halpha_sunspot_temporal_degrade.sav`
- `Halpha_sunspot_spatial_degrade.sav`
- `Halpha_sunspot_umbra_apodized.sav`
- `aia1700.sav`
- `aia171.sav`

all of which are described in the table above.

Step 03:

Move the **QUEEFF_IDL** folder to your usual IDL working directory:

```
$ mv  QUEEFF_IDL  /Users/QUB/IDL_programmes/
```

Following this, your usual IDL working directory will contain the framework necessary to run the Fourier filtering code, notably the parent **QUEEFF_IDL** folder hosting the main `QUB_QUEEFF.pro` code, the `QUEEFF_dependencies.bat` file for compiling the necessary related programmes, and the secondary **QUEEFF_dependencies** folder that contains additional (non-standard) IDL codes required for the main `QUB_QUEEFF.pro` code operation. Please ensure that this IDL working directory is compiled each time you start SSWIDL. For testing purposes, move the reference data directory to a suitable place for analysis:

```
$ mv  Reference_Data  /Users/QUB/Data/
```

Step 04:

The Fourier filtering code can be run from any location on your computer. However, it is common for the user to move into the folder containing the data prior to opening SSWIDL. Therefore, move to your analysis directory and begin an SSWIDL session, before running the `QUEEFF_dependencies.bat` check file:

```
$ cd  /Users/QUB/Data/Reference_data
$ sswidl
IDL>  @QUEEFF_dependencies.bat
```

The `QUEEFF_dependencies.bat` check file will confirm that every necessary code for the successful application of the filtering is compiled. At this point, check the terminal outputs to ensure that every module was successfully compiled. If any do not compile, check that the parent Fourier filtering folder is sourced by IDL (e.g., the **QUEEFF_IDL** folder) and that your version of SSWIDL is fully up-to-date.

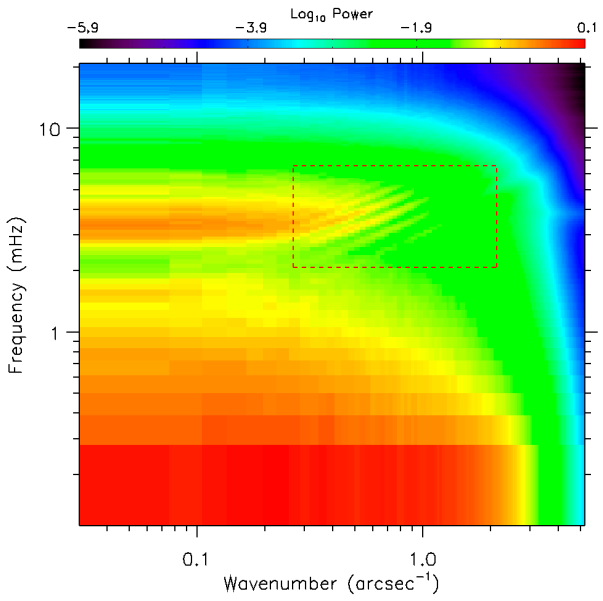

Figure 1: The $k-\omega$ diagram, where k_x and k_y are collapsed into a single wavenumber, k, generated by the `QUB_QUEEFF.pro` code for the SDO/AIA 1700 Å reference data. The dashed red box denotes the user specified region for filtering, such that all Fourier power outside the box will be suppressed in the output datacube.

Now load in the sample SDO/AIA 1700 Å time series, which will restore a variable called `dat1700`, containing the full [350, 350, 373] image cube.

```
IDL>  restore, '/Users/QUB/Data/Reference_data/aia1700.sav', /ver
```

This SDO/AIA 1700 Å datacube is the input image sequence for the `QUB_QUEEFF.pro` code, which will undertake the spatial and/or temporal Fourier filtering of the data.

Step 05 – Basic Functionality:

The `QUB_QUEEFF.pro` code allows for the submission of a number of optional keywords. The most basic call requires the input 3D datacube (i.e., [x, y, time]), spatial sampling of the data (in arcsec/pixel) and the cadence between subsequent images (in seconds). In its most basic form, the programme can be called using:

```
IDL>  filtered_cube = QUB_QUEEFF(dat1700, 0.6, 24.0)
```

where '0.6' refers to the pixel size in arcseconds and '24.0' corresponds to the time interval between successive images in the time series. *This is the most simple method of running the code.* You will first see values such as the spatial/temporal resolutions and Nyquist values displayed in the terminal. With no filter parameters specified in the calling sequence, the code will generate a $k-\omega$ diagram for your inspection, before asking the user to click on the regions of interest for filtering. To isolate the $k-\omega$ region the code will preserve, click the cursor in the lower-left corner, followed by the upper-right corner, of the rectangular region of interest in the $k-\omega$ diagram displayed on screen. The $k-\omega$ diagram will then be superimposed by a dashed red box that denotes the chosen region of interest, as seen in Figure 1.

With the filter region set in both the spatial and temporal domains, the code will begin the process of extracting the information contained within this region via Fourier techniques. Whilst the filtering process is ongoing, an additional plot will be generated displaying the spatial and temporal Fourier power spectra, alongside the effect of the filtering on the power distributions, as seen in Figure 2. When complete, the filtering will provide a new datacube (of equal dimensions to the original input) containing just the information embedded within the $k-\omega$ region of interest.

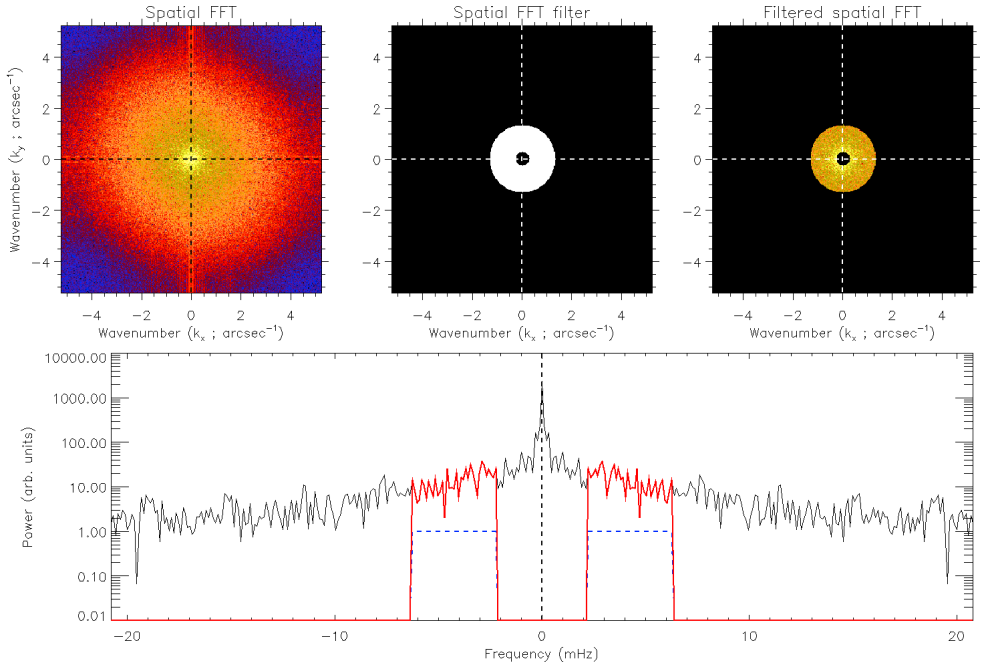

Figure 2: **Upper left panel:** The time-averaged spatial power spectrum of the dataset. **Upper central panel:** A mask depicting the chosen spatial filter. **Upper right panel:** The result of multiplying the mask by the spatial Fourier transform, revealing the spatial scales that are returned in the filtered cube. **Lower panel:** The spatially-averaged temporal power spectrum plotted in black, with the temporal filter mask overplotted using a dashed blue line, and the preserved Fourier power displayed with a solid red line.

Step 06 – Expert Functionality:

The test run described above represents the simplest configuration for the application of temporal and spatial filtering using the `QUB_QUEEFF.pro` code. However, there are utilities that allow for a more streamlined and accurate result, where the process can be tailored to the objectives of the user. Specifically, the optional keywords that allow for more precise data filtering include:

- **f1** = optional lower (temporal) frequency to filter – given in mHz
- **f2** = optional upper (temporal) frequency to filter – given in mHz
- **k1** = optional lower (spatial) wavenumber to filter – given in arcsec^{-1} (where $k = 2\pi/\lambda$)
- **k2** = optional upper (spatial) wavenumber to filter – given in arcsec^{-1} (where $k = 2\pi/\lambda$)
- **spatial_torus** = optional keyword that makes the annulus used for spatial filtering have a Gaussian-shaped profile (useful for preventing aliasing)
- **temporal_torus** = optional keyword that makes the temporal filter have a Gaussian-shaped profile (useful for preventing aliasing)
- **no_spatial_filt** = optional keyword that ensures no spatial filtering is performed on the dataset (i.e., only temporal filtering)
- **no_temporal_filt** = optional keyword that ensures no temporal filtering is performed on the dataset (i.e., only spatial filtering)

Selecting filter ranges

For instances where an exact filter region is desired, or earlier runs of the code have highlighted a region of interest in the $k-\omega$ diagram, the exact boundaries of the filter region can be defined, eliminating any user input during the running of the code. As an example, to re-create the filter region in Figure 1, call the code as follows, where the frequencies are defined in mHz, and the wavenumber in arcsec^{-1}.

```
IDL>  filtered_cube = QUB_QUEEFF(dat1700, 0.6, 24.0, f1=2.22, f2=6.30, k1=0.27, k2=1.3)
```

Correcting aliasing

In the default mode, the `QUB_QUEEFF.pro` code applies a binary filter mask, as seen in Figure 2 (i.e. the FFT is multiplied by a mask where all retained values are set to 1, and all discarded values are set to zero). Applying a hard boundary in the mask can lead to sample aliasing, where signals around the boundary become indistinguishable, leading to unwanted fringe patterns in the filtered cube. A remedy for this is to smooth the edges of the filter mask, using a 1D Gaussian for the temporal domain, and a 2D torus for the spatial domain. To apply the smoothing, the following keywords must be added during the calling of the code:

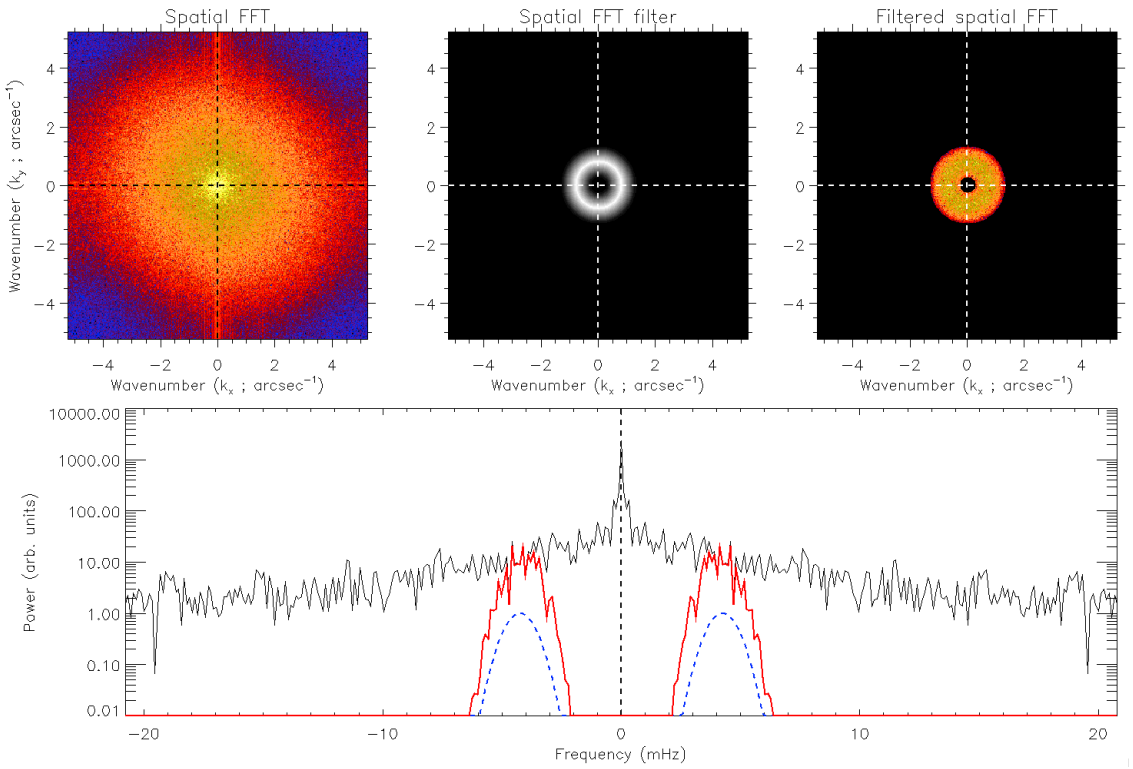

Figure 3: The filter specifications for Gaussian-smoothed temporal/spatial masks, in an equivalent format to Figure 2. The spatial filter mask (upper central panel) exhibits a gradual decrease from the centre of the toroid, and the temporal filter, seen in blue in the lower panel, is now a 1D Gaussian.

```
IDL>  filtered_cube = QUB_QUEEFF(dat1700, 0.6, 24.0, f1=2.22, f2=6.30, k1=0.27, k2=1.3,
                    /temporal_torus, /spatial_torus)
```

The specifications of the smoothing can be seen in the filter outputs presented in Figure 3, and effectively suppresses noise artefacts for all filter ranges. *Note: It is recommended that the spatial and temporal smoothing is ALWAYS applied.* In the case where a datacube is being filtered across a small range of frequencies or wavenumbers, it is viable to remove the aliasing correction, however proceed cautiously, ensuring that no fringes interact with true signals in the filtered cube.

Individual Filtering

If the user wishes to produce a traditional temporal Fourier filtered cube, or to just investigate the length scales across all frequencies inherent within the data, the `QUB_QUEEFF.pro` code can conduct either temporal or spatial filtering without the need of the other using the following keywords:

```
IDL>  filtered_cube = QUB_QUEEFF(dat1700, 0.6, 24.0, f1=2.22, f2=6.30, /no_spatial_filt)
```

OR

```
IDL>  filtered_cube = QUB_QUEEFF(dat1700, 0.6, 24.0, k1=0.27, k2=1.30, /no_temporal_filt)
```

Final Remarks:

Hopefully the `QUB_QUEEFF.pro` code will assist with all aspects of data preparation and analysis. Any problems encountered or requests for features can be directed to Dr. David Jess. As a final reminder, if you employ this code please quote the Jess et al. (2017) paper that first made use of this algorithm.

References

Jess, D. B., De Moortel, I., Mathioudakis, M., et al. 2012, *ApJ*, **757**, 160

Jess, D. B., Dillon, C. J., Kirk, M. S., et al. 2019, *ApJ*, **871**, 133

Jess, D. B., Reznikova, V. E., Ryans, R. S. I., et al. 2016, *Nature Physics*, **12**, 179

Jess, D. B., Van Doorsselaere, T., Verth, G., et al. 2017, *ApJ*, **842**, 59

Krishna Prasad, S., Jess, D. B., & Khomenko, E. 2015, *ApJL*, **812**, L15

Krishna Prasad, S., Jess, D. B., Klimchuk, J. A., & Banerjee, D. 2017, *ApJ*, **834**, 103