# INM378 PG Digital Signal Processing Coursework Report

By Jacques Leplat

## Coding Approach

All of sections make use of Python classes. This makes the implementation easier to understand.

## Controllable Filter Task

The implementation was split into Python classes:

### WavClass

Loads wav sound files, converts stereo to mono.

**convertToMono()**: The correlation coefficient of channel 1 to channel 2 is calculated. This is used to decide whether channel 1 should be inverted and/or the phase adjusted.

**alignSamples()** implements the inverting and search for the optimal correlation coefficient to adjust the phase of channel 1.

RockA.wav was found to have a negative and out of phase channel:

|  |  |
| --- | --- |
| Figure RockA.wav mono conversion with no alignment | Figure Correlation Coefficient Search |

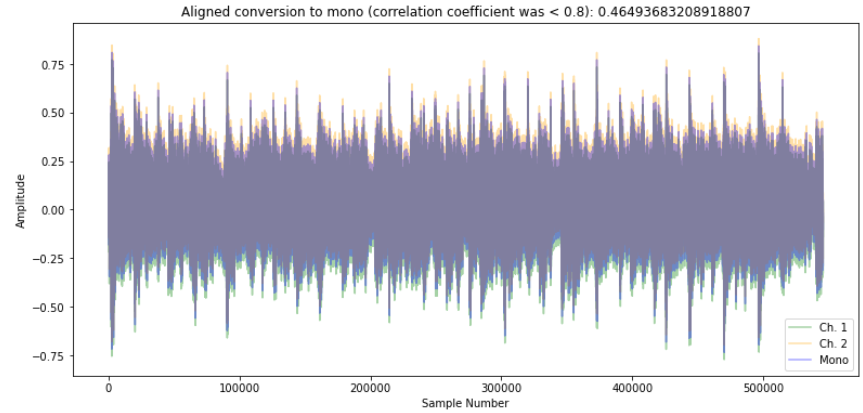


Figure Correlated Mono Conversion

Figure 1 shows adding (channel 1 and 2) / 2 creates a poor (purple mono conversion). Figure 2 shows the amount to shift the channel 1 so that is in phase with channel 2.

### ControlClass

Implements a control signal to interpolate between two filters. A sample rate, number of samples, and frequency in Hertz is supplied to generate a repeating sine control signal that has the same length as the number of samples.

|  |  |
| --- | --- |
| Figure Test control signal | Figure Test Impulse Response plots |

A test control class is created (figure 4). The resulting modulated filter for samples 0, 32 and 63 is shown in figure 5.

### ConvolutionClass

Performs the convolution of a signal with a filter. The convolution implements modulation with a control signal.

(Smith, 1999) Chapter 6, explains input side and output side convolution. Both methods are implemented:

convolveInputSide(signal). Input side convolution requires the complete signal to be present.

convolveOutputSide(signal). The complete input signal is not necessary.

Both implementations call on the control class to get the interpolated filter array for the requested sample. This filter, also referred to as kernel or filter kernel (Smith, 1999), it is the impulse response of the filter.

## Frequency Response and Impulse Response

The filters which are interpolated are comb filters. In the time spectrum, two filters are defined. The filters are small (64 values) so that the convolution is completed fast when applied to a large input signal, the CombFilterClass instance is created as shown below:

filter = CombFilterClass(controlClass, minDelay=4, maxDelay=32, filterSize=64, doPlot=True)

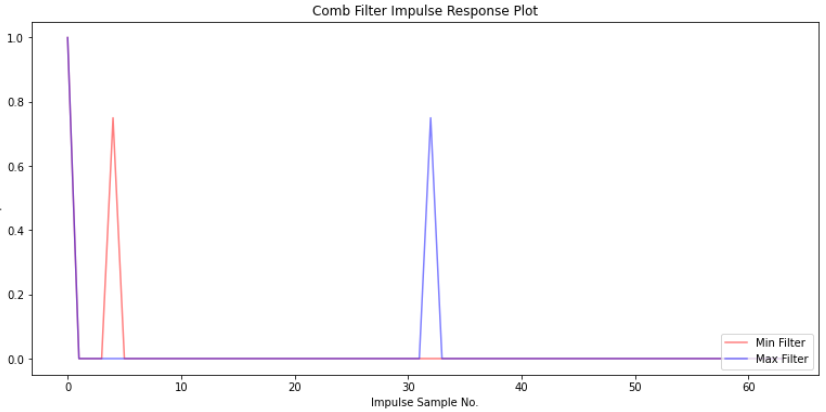


Figure Impulse response of the comb filters

When convolving an impulse signal with the above filter, the initial value of 10 at sample 0 also produces a new sample of 8 at position 32.

Convolution in the time domain is the same as multiplication at the frequency domain. The above filter in the frequency domain is shown below:

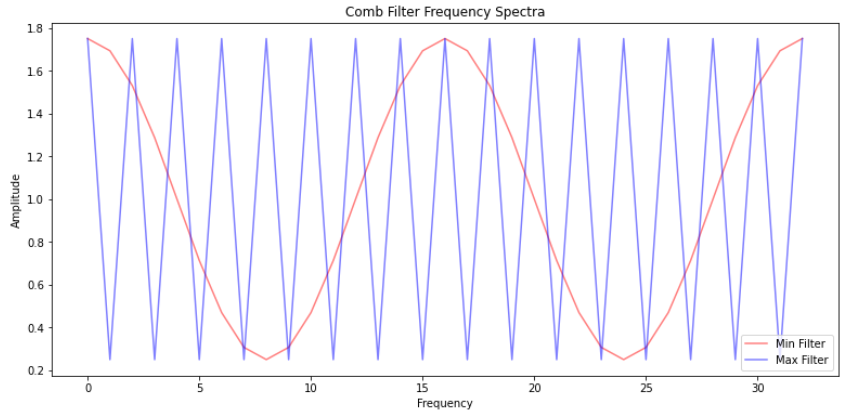


Figure Frequency Response of the Comb Filters

Figure 7 illustrates the range of frequencies that are allowed through (when the amplitude is 1), or not allowed through (when the amplitude is at its lowest). When applied to an input signal, the filtering is repeated for all frequencies. This filtering is achieved by multiplication of the frequency response of the filter by the Fourier Transform of the signal: since the amplitude is not lower than zero, and not larger than one, multiplication of the signal’s frequency domain has the effect of either leaving a frequency unchanged, or reduced in amplitude.

As can be seen in the spectrogram below of the modulated (2Hz) filter on carrier.wav, produces the following changes to the original signal:

|  |  |
| --- | --- |
| Figure carrier.wav Spectrogram | Figure carrier.wav after filtering |

The modulated filter oscillates between the minimum filtering of frequencies and maximum filtering of frequencies, seen in the frequency response of the filters (Figure 7).

## Digit Recognition

The digit recognition task was implemented with Python classes.

### ImageClass

Loads a set of images and their labels (test or train) using readImages().

displayLabelsStatitics(), Figure 10 below: the test and train images form balanced, randomly distributed datasets.

displayImages() displays several images:

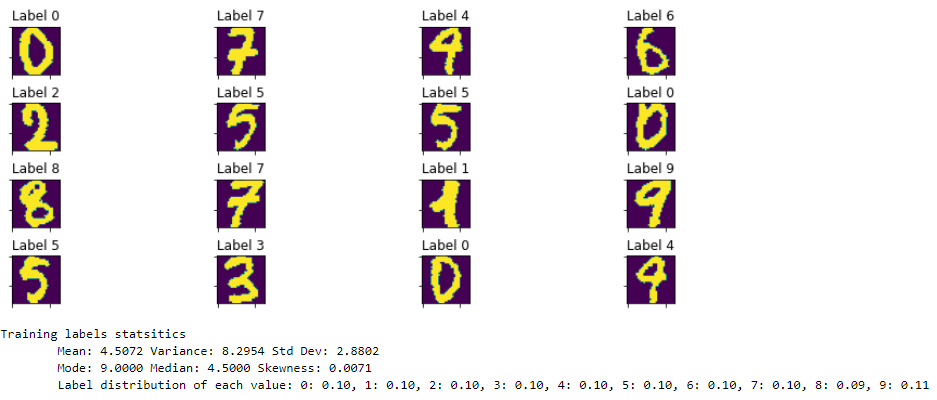


Figure Images and Label Statistics

Methods are implemented to : create negative versions, rotate images, add noise to the images, offset the images, and copy image class instances.

|  |  |
| --- | --- |
| Figure All effects | The images are rotated by 15 degrees, offset 5 pixels to the right, and 1 pixel down, as negatives and noise is added. |

### ImageCorrelationClass

Performs simple correlation and 2d correlation of images, calculates the accuracy of the correlations, and compares the correlationaccuracies.

|  |  |  |
| --- | --- | --- |
| **Test Training vs Test Images** | **Simple Correlation** | **2d Correlation** |
| All training images, All Test Images | 0.85 | --- |
| Negative 100 training, negative 100 test | 0.93 | 0.93 |
| 100 training rotated, 100 test | 0.68 | 0.71 |
| 100 training noisy, 100 test | 0.89 | 0.89 |
| 100 training offset, 100 test | 0.27 | 0.91 |
| 100 training rotated, noisy, 100 test | 0.66 | 0.69 |
| 100 training rotated, negative, noisy, 100 negative test | 0.69 | 0.69 |
| 100 training rotated, negative, noisy, offset, 100 negative test | 0.18 | 0.18 |

Figure Correlation Accuracy Comparison

The last comparison is unexpected for 2d correlation. 2d correlation is performed: the test for offsetting images accuracies is 0.27 vs 0.91 for 2d correlation.

## Time Series Prediction with Financial Data

FinancialDataClass2 is implemented to wrap the operations for tasks 1 – 3: load the data, clean the data, plot the data and select a column of interest (Real\_Price). De-trending, and seeking long term trends using linear fit using and log10 is also present in this class using the method finClass.deTrend("Real\_Price").

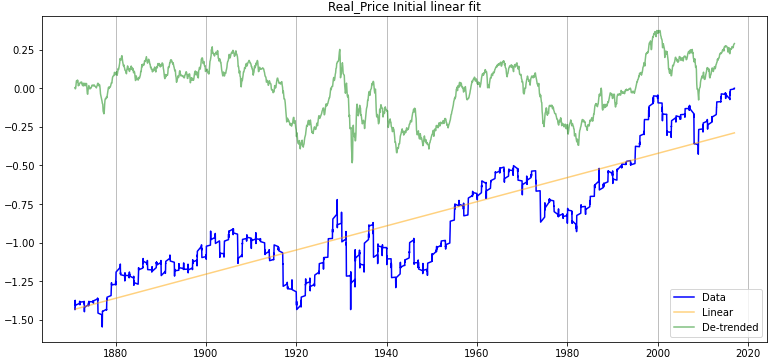


Figure Log10, single linear fit

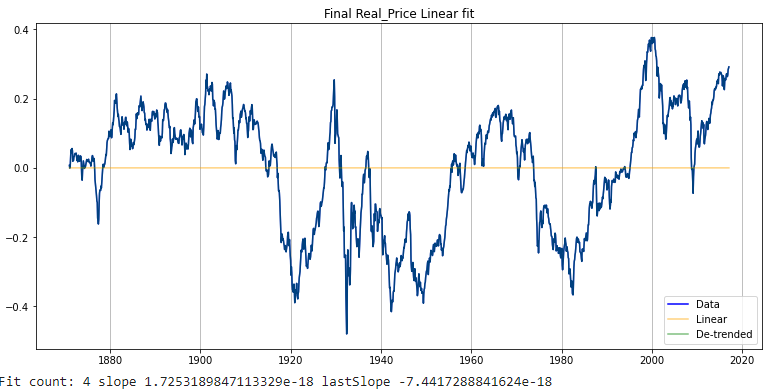
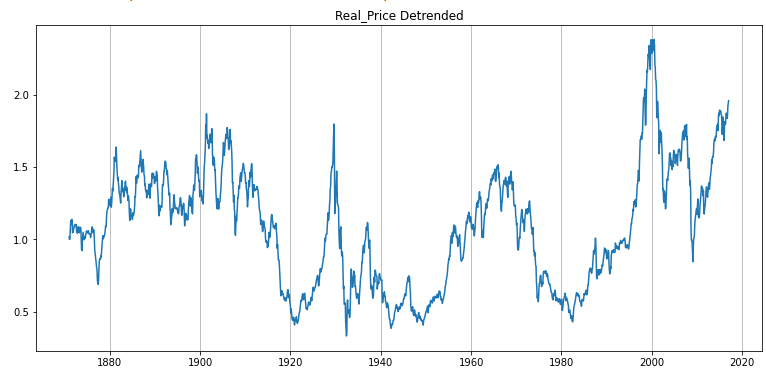


Figure Log10, 4 linear fit calls



The long term trend of the Real\_Price is upwards, but there are some significant drops that take several years to recover.

### FinancialFft Class

Financial Fft class is used to perform a Fourrier analysis of the data.

A spectrogram of the de-trended data is used to look for periodicity:

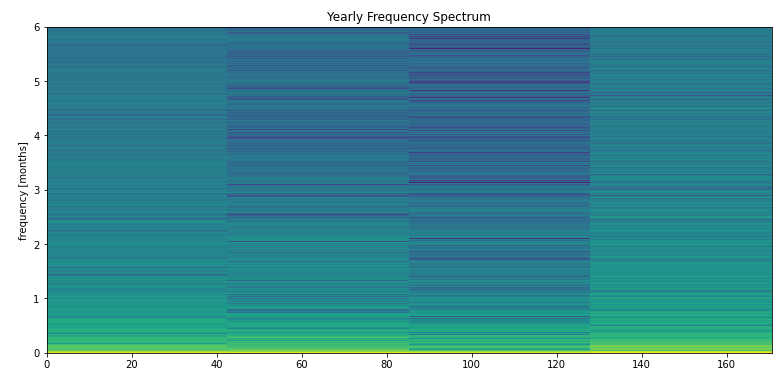


Figure SPectrogram of de-trended Real\_Price

The spectrogram of De-Trended Real\_Price shows a periodicity: (1871/1 - 1921/8) 608 months, (1921/9 - 1966/1) 533 months, (1966/2 - 2000/08) 415 months.

Each period is used to see if the predictions are useful. Below is the one for 608 months (1871/1 - 1921/8):

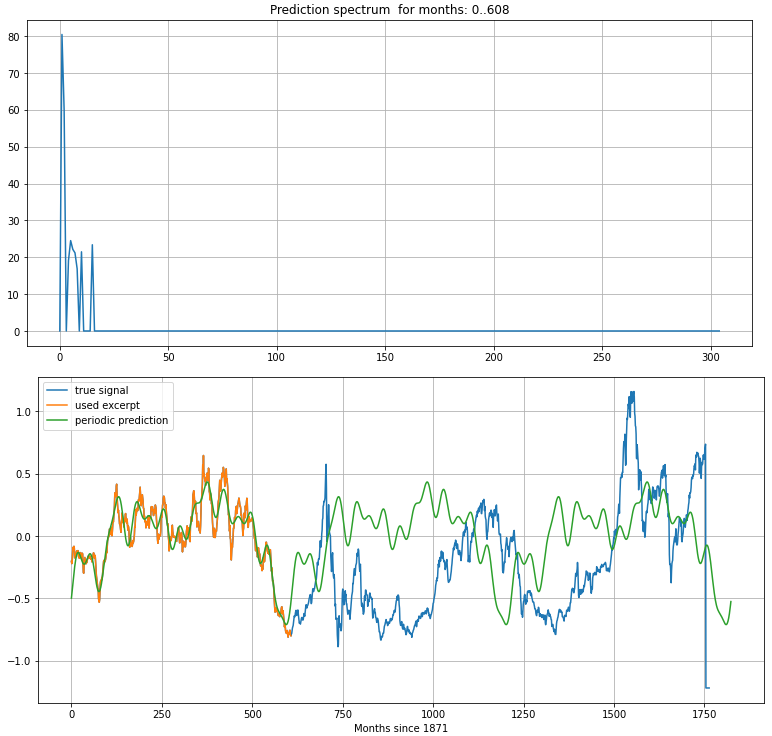


Figure Harmonic spectrum chosen and resulting prediction

### Conclusion

### The Fourier transform is very good at highlighting periodicities in data, as seen in the spectrogram. It can be used to identify the start and end of price cycles (Stádník et al., 2016).

The Fourier Transform is not suited to share price prediction using periodicity. This is due to the unpredictability of price changes.

## References

Smith, S.W., 1999. The scientist and engineer’s guide to digital signal processing. California Technical Pub., San Diego, Calif.

Stádník, B., Raudeliūnienė, J., Davidavičienė, V., 2016. FOURIER ANALYSIS FOR STOCK PRICE FORECASTING: ASSUMPTION AND EVIDENCE. J. Bus. Econ. Manag. 17, 365–380. https://doi.org/10.3846/16111699.2016.1184180