

03_kalman_filter_and_wavelets

September 29, 2021

1 How to use the Kalman Filter and Wavelets

1.1 Imports & Settings

```
[1]: import warnings
      warnings.filterwarnings('ignore')
```

```
[2]: %matplotlib inline

      from datetime import datetime
      import itertools

      import pandas as pd
      import pandas_datareader.data as web
      from pykalman import KalmanFilter
      import pywt

      import matplotlib.pyplot as plt
      import seaborn as sns
```

```
[3]: warnings.filterwarnings('ignore')
      sns.set_style('whitegrid')
      idx = pd.IndexSlice
```

1.2 Get Data

The `assets.h5` store can be generated using the the notebook [create_datasets](#) in the [data](#) directory in the root directory of this repo for instruction to download the following dataset.

We load the Quandl stock price datasets covering the US equity markets 2000-18 using `pd.IndexSlice` to perform a slice operation on the `pd.MultiIndex`, select the adjusted close price and unpivot the column to convert the DataFrame to wide format with tickers in the columns and timestamps in the rows:

Set data store location:

```
[4]: DATA_STORE = '../data/assets.h5'
```

```
[5]: with pd.HDFStore(DATA_STORE) as store:
      sp500 = store['sp500/stooq'].loc['2009': '2010', 'close']
```

1.3 Kalman Filter

1.3.1 Configuration

```
[6]: kf = KalmanFilter(transition_matrices = [1],
                       observation_matrices = [1],
                       initial_state_mean = 0,
                       initial_state_covariance = 1,
                       observation_covariance=1,
                       transition_covariance=.01)
```

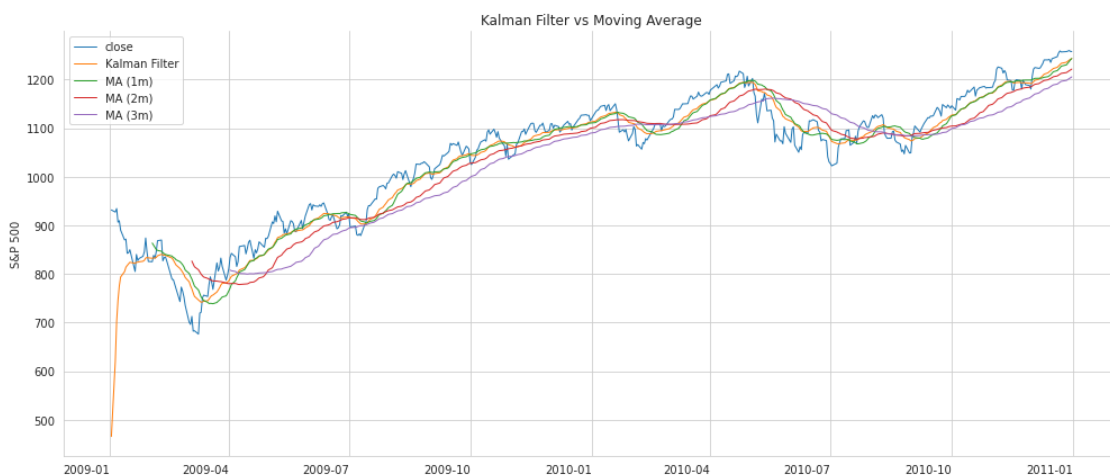
1.3.2 Estimate the hidden state

```
[7]: state_means, _ = kf.filter(sp500)
```

1.3.3 Compare with moving average

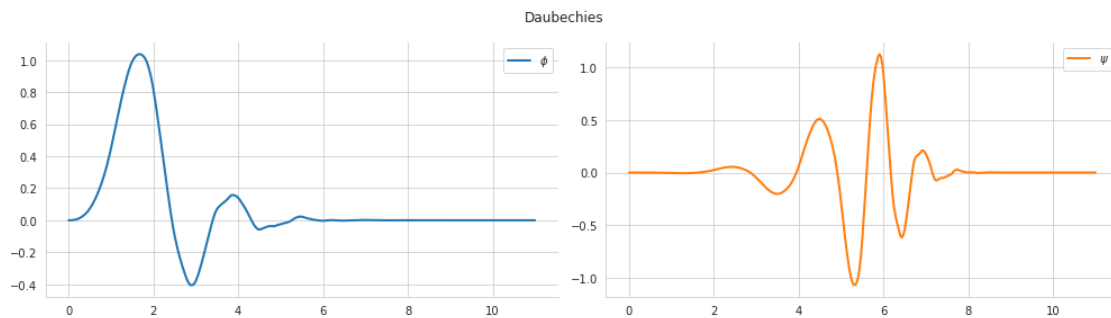
```
[8]: sp500_smoothed = sp500.to_frame('close')
      sp500_smoothed['Kalman Filter'] = state_means
      for months in [1,2,3]:
          sp500_smoothed[f'MA ({months}m)'] = sp500.rolling(window=months*21).mean()

      ax = sp500_smoothed.plot(title='Kalman Filter vs Moving Average',
                               figsize=(14,6), lw=1, rot=0)
      ax.set_xlabel('')
      ax.set_ylabel('S&P 500')
      plt.tight_layout()
      sns.despine();
```



1.4 Wavelets

```
[9]: wavelet = pywt.Wavelet('db6')
phi, psi, x = wavelet.wavefun(level=5)
df = pd.DataFrame({' $\phi$ ': phi, ' $\psi$ ': psi}, index=x)
df.plot(title='Daubechies', subplots=True, layout=(1, 2), figsize=(14, 4),
        lw=2, rot=0)
plt.tight_layout()
sns.despine();
```



```
[10]: plot_data = [('db', (4, 3)),
                   ('sym', (4, 3)),
                   ('coif', (3, 2))]

for family, (rows, cols) in plot_data:
    fig = plt.figure(figsize=(24, 12))
    fig.subplots_adjust(hspace=0.2, wspace=0.2, bottom=.02, left=.06,
                        right=.97, top=.94)
    colors = itertools.cycle('bgrcmk')

    wnames = pywt.wavelist(family)
    i = iter(wnames)
    for col in range(cols):
        for row in range(rows):
            try:
                wavelet = pywt.Wavelet(next(i))
            except StopIteration:
                break
            phi, psi, x = wavelet.wavefun(level=5)

            color = next(colors)
```

```

        ax = fig.add_subplot(rows, 2 * cols, 1 + 2 * (col + row * cols))
        ax.set_title(wavelet.name + " phi")
        ax.plot(x, phi, color, lw=1)
        ax.set_xlim(min(x), max(x))

        ax = fig.add_subplot(rows, 2*cols, 1 + 2*(col + row*cols) + 1)
        ax.set_title(wavelet.name + " psi")
        ax.plot(x, psi, color, lw=1)
        ax.set_xlim(min(x), max(x))
    sns.despine()

for family, (rows, cols) in [ ('bior', (4, 3)), ('rbio', (4, 3))]:
    fig = plt.figure(figsize=(24, 12))
    fig.subplots_adjust(hspace=0.5, wspace=0.2, bottom=.02, left=.06,
                        right=.97, top=.94)

    colors = itertools.cycle('bgrcmk')
    wnames = pywt.wavelist(family)
    i = iter(wnames)
    for col in range(cols):
        for row in range(rows):
            try:
                wavelet = pywt.Wavelet(next(i))
            except StopIteration:
                break
            phi, psi, phi_r, psi_r, x = wavelet.wavefun(level=5)
            row *= 2

            color = next(colors)
            ax = fig.add_subplot(2*rows, 2*cols, 1 + 2*(col + row*cols))
            ax.set_title(wavelet.name + " phi")
            ax.plot(x, phi, color, lw=1)
            ax.set_xlim(min(x), max(x))

            ax = fig.add_subplot(2*rows, 2*cols, 2*(1 + col + row*cols))
            ax.set_title(wavelet.name + " psi")
            ax.plot(x, psi, color, lw=1)
            ax.set_xlim(min(x), max(x))

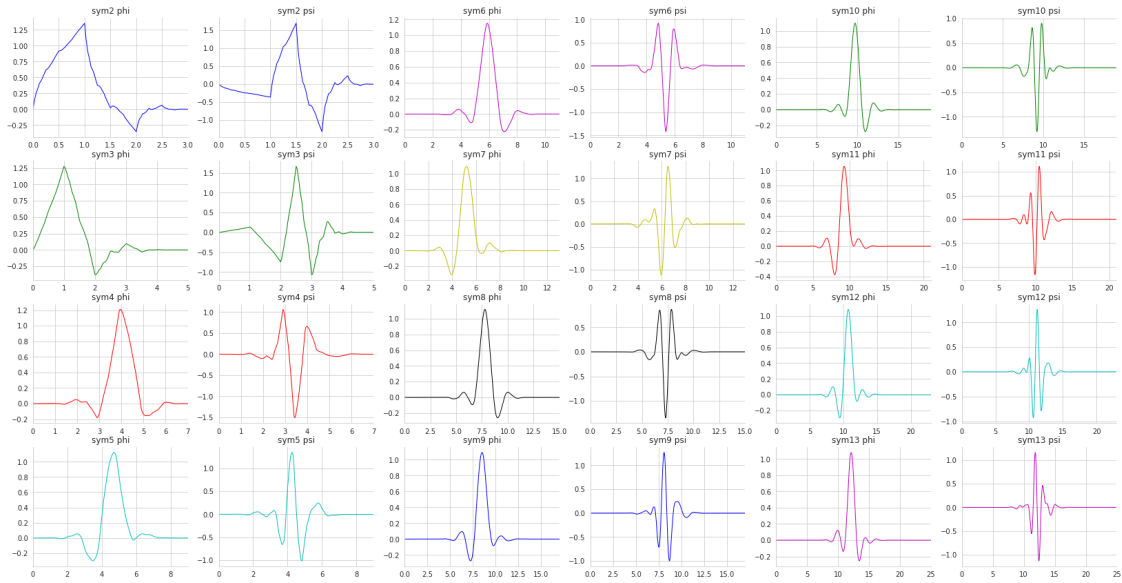
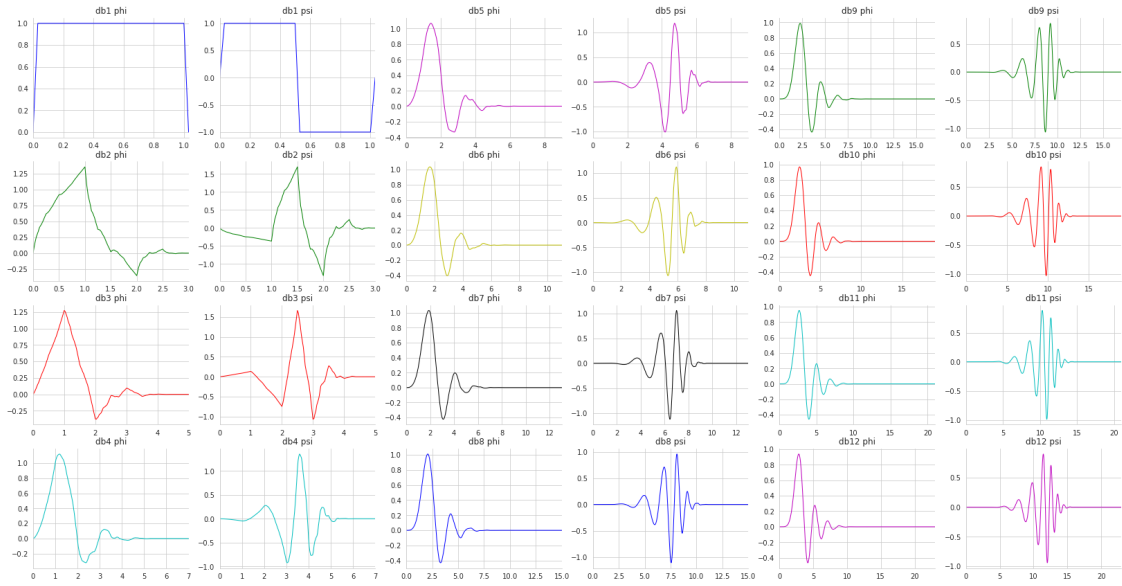
            row += 1
            ax = fig.add_subplot(2*rows, 2*cols, 1 + 2*(col + row*cols))
            ax.set_title(wavelet.name + " phi_r")
            ax.plot(x, phi_r, color, lw=1)
            ax.set_xlim(min(x), max(x))

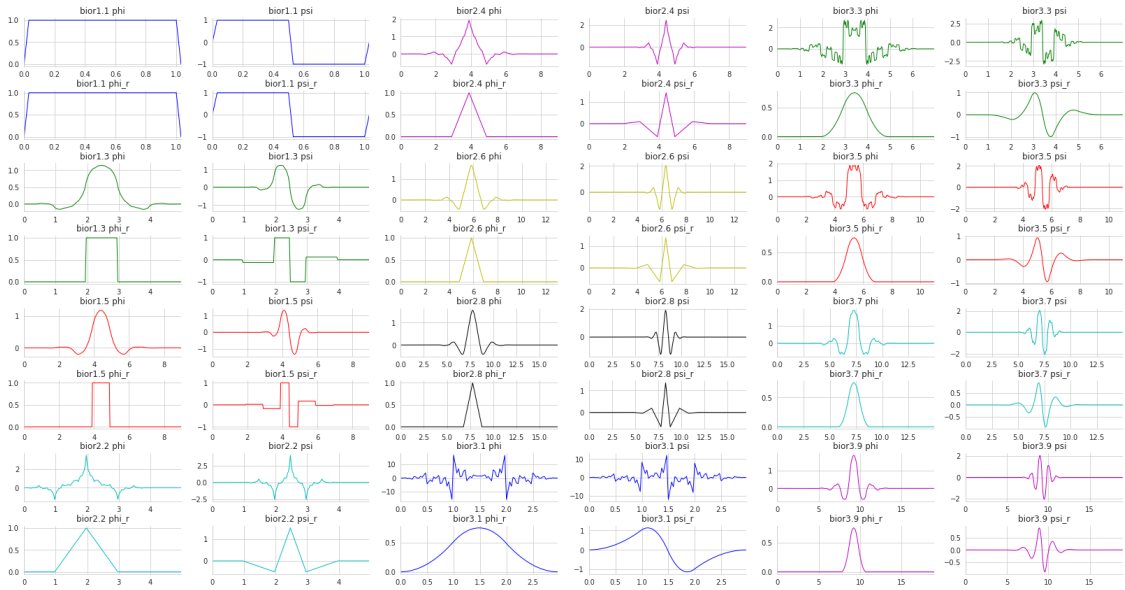
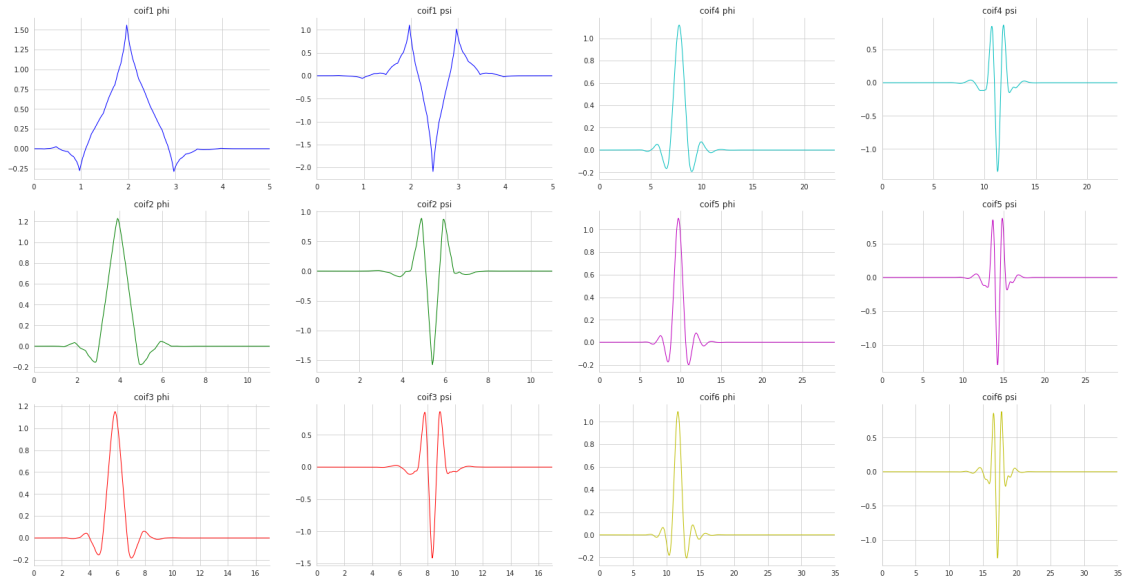
            ax = fig.add_subplot(2*rows, 2*cols, 1 + 2*(col + row*cols) + 1)
            ax.set_title(wavelet.name + " psi_r")

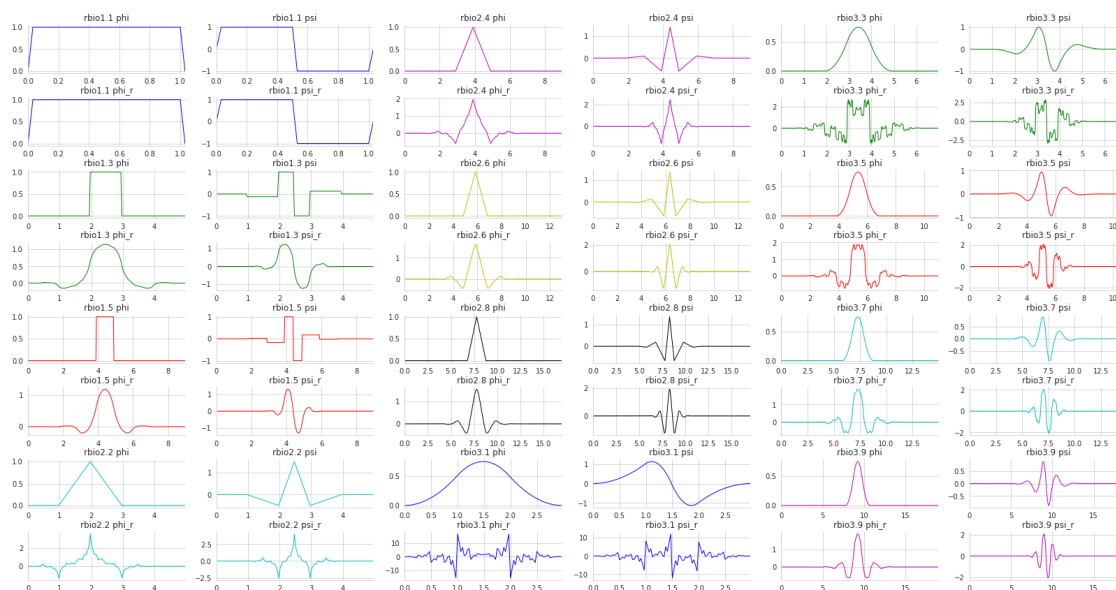
```

```
ax.plot(x, psi_r, color, lw=1)
ax.set_xlim(min(x), max(x))
sns.despine()
```

```
plt.show()
```







1.4.1 Available Wavelet Families

```
[11]: pywt.families(short=False)
```

```
[11]: ['Haar',
       'Daubechies',
       'Symlets',
       'Coiflets',
       'Biorthogonal',
       'Reverse biorthogonal',
       'Discrete Meyer (FIR Approximation)',
       'Gaussian',
       'Mexican hat wavelet',
       'Morlet wavelet',
       'Complex Gaussian wavelets',
       'Shannon wavelets',
       'Frequency B-Spline wavelets',
       'Complex Morlet wavelets']
```

1.4.2 Smoothing the S&P 500

```
[12]: signal = (pd.read_hdf(DATA_STORE, 'sp500/stooq')
                .loc['2008': '2009']
                .close.pct_change()
                .dropna())

fig, axes = plt.subplots(ncols=2, figsize=(14, 5))
```

```

wavelet = "db6"
for i, scale in enumerate([.1, .5]):

    coefficients = pywt.wavedec(signal, wavelet, mode='per')
    coefficients[1:] = [pywt.threshold(i, value=scale*signal.max(),
    ↪mode='soft') for i in coefficients[1:]]
    reconstructed_signal = pywt.waverec(coefficients, wavelet, mode='per')
    signal.plot(color="b", alpha=0.5, label='original signal', lw=2,
               title=f'Threshold Scale: {scale:.1f}', ax=axes[i])
    pd.Series(reconstructed_signal, index=signal.index).plot(c='k', label='DWT
    ↪smoothing}', linewidth=1, ax=axes[i])
    axes[i].legend()
fig.tight_layout()
sns.despine();

```

