Topics in Time Series Analysis: Assignment 1

Lucas Cruz Fernandez

1544674, lcruzfer@mail.uni-mannheim.de

04th of May 2020

Q13

We want to find the coefficients of the filter described in the assignment. Using the hint, we know that the coefficients c_j can be written as:

$$c_j = \frac{1}{2\pi} \int_{-pi}^{pi} e^{ij\lambda} \psi(\lambda) d\lambda$$

We can insert the transfer function of the ideal band-pass filter by splitting up the integral and writing:

$$2\pi c_j = \int_{-\pi}^{-\lambda_h} e^{ij\lambda} \times 0 d\lambda + \int_{-\lambda_h}^{-\lambda_l} e^{ij\lambda} \times 1 d\lambda + \int_{-\lambda_l}^{\lambda_l} e^{ij\lambda} \times 0 d\lambda + \int_{\lambda_l}^{\lambda_h} e^{ij\lambda} \times 1 d\lambda + \int_{\lambda_h}^{\pi} e^{ij\lambda} \times 0 d\lambda$$

Splitting up the integral allows us plugging in the transfer function $\psi(\lambda)$ with its respective value for the respective limits.

Now further rearranging we can show that:

$$\begin{aligned}
& 2\pi c_j = \int_{-\lambda_h}^{-\lambda_l} e^{ij\lambda} d\lambda + \int_{\lambda_l}^{\lambda_h} e^{ij\lambda} d\lambda \\
\Leftrightarrow \quad & 2\pi c_j = \left[\frac{1}{ij} e^{ij\lambda}\right]_{-\lambda_h}^{-\lambda_l} + \left[\frac{1}{ij} e^{ij\lambda}\right]_{\lambda_l}^{\lambda_h} \\
\Leftrightarrow \quad & 2\pi c_j = \frac{1}{ij}\left[e^{-ij\lambda_l} - e^{-ij\lambda_h} + e^{ij\lambda_h} - e^{ij\lambda_l}\right] \\
\Leftrightarrow \quad & 2\pi c_j = \frac{1}{ij}\left[e^{-ij\lambda_l} - e^{ij\lambda_l} + e^{ij\lambda_h} - e^{-ij\lambda_h}\right] \\
\Leftrightarrow \quad & 2\pi c_j = \frac{1}{ij}\left[-2i\sin(j\lambda_l) + 2i\sin(j\lambda_h)\right] \\
\Leftrightarrow \quad & c_j = \frac{\sin j\lambda_h - \sin j\lambda_l}{j\pi}
\end{aligned}$$

However, for c_0 the solution must be derived again by setting $j = 0$ beforehand (the above term has no solution for $j = 0$).

$$\begin{aligned}
& 2\pi c_j = \int_{-\lambda_h}^{-\lambda_l} e^{i0\lambda} d\lambda + \int_{\lambda_l}^{\lambda_h} e^{i0\lambda} d\lambda \\
\Leftrightarrow \quad & 2\pi c_j = \int_{-\lambda_h}^{-\lambda_l} d\lambda + \int_{\lambda_l}^{\lambda_h} d\lambda \\
\Leftrightarrow \quad & 2\pi c_j = [-\lambda_l + \lambda_h + \lambda_h - \lambda_l] \\
& c_j = \frac{\lambda_h - \lambda_l}{\pi}
\end{aligned}$$

Q14

Considering the given process we can show that its transfer function can be written as

$$\psi(\lambda) = \frac{1}{2q+1} \sum_{|j| \le q} e^{-ij\lambda}$$

and we therefore know that the spectral density function is:

$$\begin{aligned}
f(\lambda) &= \frac{1}{(2q+1)^2} \sum_{|m| \le q} e^{-ij\lambda} \sum_{|k| \le q} e^{ij\lambda} \\
&= \frac{1}{(2q+1)^2} \sum_{|m| \le q} \sum_{|k| \le q} e^{(k-m)ij\lambda}
\end{aligned}$$

We can easily code this function and plot it accordingly for $\lambda \in [0, \pi]$.
First, importing necessary modules for Python, before we define the functions we will need throughout this task. Since the spectral density returns a complex number for some values of $\lambda \in [0, \lambda]$.

```
    import numpy as np
    from numpy import random as rd
    import pandas as pd
    import scipy.integrate as integrate
    import matplotlib.pyplot as plt
#defining the spectral density function
    def spec_dens(lam, sigma = 1, q = 3):
        double_sum = 0
        for k in range(-q,q+1):
            for i in range(-q,q+1):
                power = (i-k)*1j*lam
                e_term = np.exp(power)
                double_sum = double_sum + e_term
        f = sigma/(2*np.pi) * 1/((2*q +1)**2) * (double_sum)
        return(f)
#defining function to get the Euclidean Norm
    def euclid_norm(complex):
        real = complex.real #returns real part of a complex number
        imag = complex.imag
        norm = np.sqrt(real**2 + imag **2)
        return(norm)
#lastly, function of s, using built in "integrate" function of scipy module
    def s(lambda_0, sigma, q):
        integral = integrate.quad(spec_dens, 0, lambda_0, args = (sigma, q))
            #integrate.quad() is standard integration
        value = 2*integral[0]
        return(value)
```

Now that the functions are ready we can define a grid of values for $\lambda \in [0, \pi]$ for which we compute the respective values of $f(\lambda)$. We also set the list of q values we are interested in $(\{1, 3, 10\})$.

```
#defining the lambda grid
    lambdas = np.arange(0, np.pi, 0.001) #from 0 to pi in steps of 0.001
#defining list of q
    qs = [1, 3, 10]
```

We set up an empty array that has the dimensions $len(qs) \times len(lambdas)$. This returns a array of arrays type of structure in which we fill each array $i \times len(lambdas)$ with the respective spectral densities of q (first array refers to $f(\lambda; q = 1)$ and so on). We then fill up this array using Python's enumerate iterator. How it works is desribed in the code comments. We will use this structure for all functions in the following applications.

```
    spec_densities = np.empty((len(qs), len(lambdas)))
    for i, q in enumerate(qs):
#enumerate creates a tuple (i, list[i]) from supplied list and then iterates of
    these pairs
        spec_densities[i] = [spec_dens(lam = l, q = q) for l in lambdas]
#applying the function spec_dens for each value in our grid of lambdas
```

We can then plot each of these 3 arrays we end up with.

```
    f = plt.figure()
    plt.grid()
#plotting the lines
    for i in range(len(results)):
        label = 'q = ' + str(qs[i])
        plt.plot(lambdas, spec_dens_norm[i], label = label)
    plt.xlabel(r'$\lambda$')
    plt.ylabel(r'$f_x(\lambda)$')
    plt.legend()
    plt.show()
```

This creates the following plot.

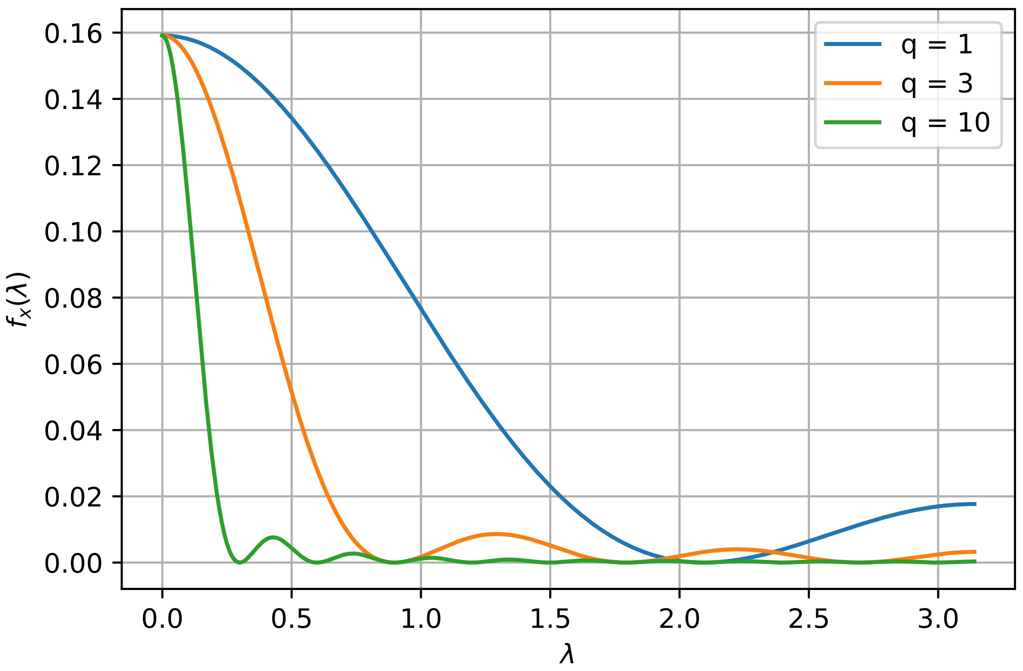

Figure 1: Spectral densities $f(\lambda; q)$ f0r $q = 1, 3, 10$

We now turn to the second plot of the assignment, plotting $VR(\lambda_0; q)$. First, we need to calculate the values of $s(\lambda_0; q)$ for $q = 1, 3, 10$ on our grid of λ_0, which is the same as the

λ grid from before. We use the same structure as before.

```
    results = np.empty((3, len(lambdas)))
    for i, q in enumerate(qs):
        results_q = [s(lam0, sigma = 1, q = q) for lam0 in lambdas]
        results[i] = results_q
```

The last element of the array $results[i]$ contains the value for $s(\pi; q)$ for $q = qs[i]$, where qs is our list of q values. Therefore, by dividing the whole array by its last element will return $VR(\lambda_0; q)$.

```
#get last element of array and divide each element of array by it
#do this for all arrays in "array of arrays" results
    VR_q = [results[i]/results[i][-1] for i in range(len(results))]
```

We can then again plot this.

```
    f2 = plt.figure()
    plt.grid()
#plotting the different VR(lam0)
    for i in range(len(results)):
        label = r'$VR(\lambda_0; q = $' + str(qs[i]) + ')'
        plt.plot(lambdas, VR_q[i], label = label)
    plt.xlabel(r'$\lambda$')
    plt.ylabel(r'$VR(\lambda_0; q)$')
    plt.legend()
    plt.show()
```

This returns Figure 2.

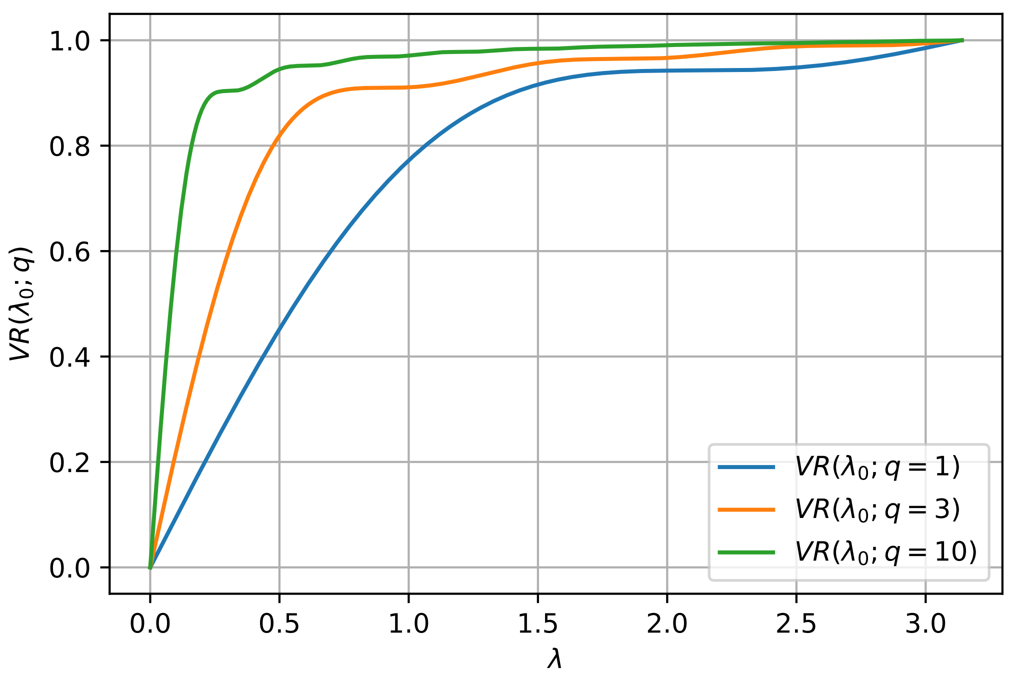

Figure 2: $VR(\lambda_0; q)$ for $q = \{1, 3, 10\}$