Skills Class Signal & Image Processing – Convolution and filtering

Joël Karel & Pietro Bonizzi

Filtering signals

Look at the documentation of the Matlab function *filter*. This function takes vectors `b`, `a`, and `x` as an input. The vectors `b` and `a` are the coefficients of polynomials in z^{-1}. It is assumed that $a(0) = 1$ (Starting with index 0 instead of 1 as in Matlab), otherwise the vectors `a` and `b` are rescaled to meet that assumption. The filter function then filters the signal `x` with the system $H(z)$:

$$H(z) = \frac{b(0) + b(1)z^{-1} + \ldots + b(N)z^{-N}}{1 + a(1)z^{-1} + \ldots + a(N)z^{-N}} \tag{1}$$

Question 3.1

Load the signal in `dataset3.mat`. Filter the signal with a moving average filter with length 3 ($b = [1/3, 1/3, 1/3]$). Verify by hand for the first six samples that the filter did indeed what you expected.

Question 3.2

Plot the filtered signal of question 3.1. The filtered signal still looks noisy. Filter the signal with moving average filters of lengths 5, 10, 20, 30 and 50. What do you observe and why?

Question 3.3

Load the signal in `data.mat`. The signal `data` has been sampled at 102.4Hz. Additionally it has been corrupted with a 50Hz noise and saved in `d1`. Design a filter with two poles and two zeros so that the 50Hz noise is filtered out by appropriately placing the poles and zeros in the z-domain. In order to do this:

- Calculate where the 50Hz and -50Hz frequencies are located in the (complex) z-domain
- Place a zero at each of these locations
- Realize that a second degree polynomial in z with as roots these two zeros is the numerator polynomial of your filter. Find this polynomial and call its coefficients b. The zeros have a strong connection so that should help.
- The poles are between $z = 0$ and the zeros. Make their magnitude 0.9 times the magnitude of the zeros, find the denominator polynomial in a similar fashion and call its coefficients a.

Inspect your filter with *freqz*, then use your filter to filter the corrupted signal `d1` and compare your filtered signal, the corrupted signal and the original data. Try it by only using a FIR filter (the coefficients b only) and the using a IIR filter (the coefficients b and a). What are the differences? Why?

Question 3.4

Suppose that someone advises you to use a low-pass filter with filter coefficients $h = [0.25, 0.5, 0.25]$. Use the *filter* function to use this on `d1` from the previous question, yielding `y1`. You think this is a good case to test the convolution theorem:

- Analytically find the frequency response $H(\omega)$ of h.
- Additionally compute (using Matlab) the FFT of `d1` denoted $F(\omega)$. Compute $Y_2(\omega) = H(\omega)F(\omega)$.
- Plot the absolute values of $F(\omega)$, $H(\omega)$, $Y_2(\omega)$ and $Y_1(\omega)$ in a single figure. As you already guessed $Y_1(\omega)$ is the FFT of `y1`. You might wanna rescale $H(\omega)$ to make it visible.
- Additionally compute `y2` and plot `y1`, `y2`, `data`, and `d1` in a single figure.

Does the convolution theorem check out?

Filtering images

Matlab presents a large variety of different image filtering functions. The Matlab filtering function *filter2* performs general 2D filtering and is a standard function shipped with Matlab. This function takes a filter-kernel `h` and an image `Im`.

Question 3.5

Read the help file of *filter2* and try:

```
>> Im = [zeros(5,3),ones(5,2)]
>> h = ones(3,3)
```

Now, `Im` is a (very small) 2D image and `h` is the filter kernel. The result of filtering `Im` with `h` is:

```
>> g = filter2(h,f);
```

Verify the result by hand

Question 3.6

Load and display the image `eight.tif`. Add *salt & pepper* noise to the image by the use of the function `noise.m`, using 0.02 as noise density. Display the result. What this noise is characterized from?

Now you have to restore the original image:

- Create a (3×3), (5×5) and a (15×15) mean kernel.
- Filter the image using *filter2*.
- Look at the results.

What happens to the noise in the images? Is a mean filter suitable for removing *salt & pepper* noise?

Question 3.7

Repeat exercise 3.4, but now by adding *additive Gaussian* noise to the original image with a 7% incidence. What happens to the noise? What happens to the image?

In the following exercise we will go into further details of the filtering process, and look at the different boundary conditions available in the function *filter2* (*same*, *valid*, and *full*):

- *same* means that the filtered image has the same dimensions as the original image.
- *valid* means that the filtered image is not affected by border problem.
- *full* means that the filtered image contained zero-padded borders.

Question 3.8

Read and display the image `circle.bmp`. Create a simple kernel for a mean filter by:

```
>> fsize = 5;
>> h = ones(fsize)/fsize^2;
```

Use *filter2* to apply the kernel to the image. Store the result into a variable called `meanim1` and display both images in the same figure by using:

```
>> figure
>> subplot(1,2,1);
>> imshow(im1,[]), colormap gray, axis image off;
>> title(Original image)
>> subplot(1,2,2);
>> imshow(meanim1,[]), colormap gray, axis image off;
>> title(Filtered image, mean filter)
```

Try *filter2* with the three different options *same*, *valid*, and *full*.