

Course material

Course:

Digital Signal Processing II

Video:

2.1.2.a The moving average filter

Concepts (extracted from automatically generated subtitles):

Local average. Average of capital m points. Simple two-point average of the current sample. Impulse response. Moving average. Smooth curve of the first panel. Capital m times. Discrete time signal. Filtered output. Two-point average. Sample of the input sequence. Non-zero term. Smoothing effect of the filter. N minus capital m . Measured value.



[to video sequence search](#)
(within Digital Signal Processing II.)

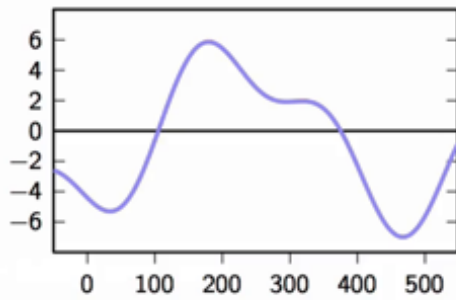


[to video](#)

Center for Digital Education. More educational support material here:

<https://exoset.epfl.ch/booc/>

Typical filtering scenario: denoising



Suppose you're taking measurements of a physical quantity that you know to be smooth and slowly varying. For instance, the tide level at the sea or the blood pressure of a patient. Because of the measurement process, your values, the values that you actually measure, are

notes

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

summary

0m 0s



.....

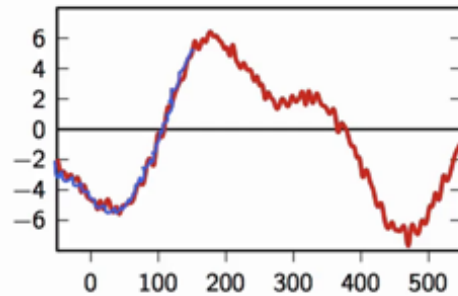
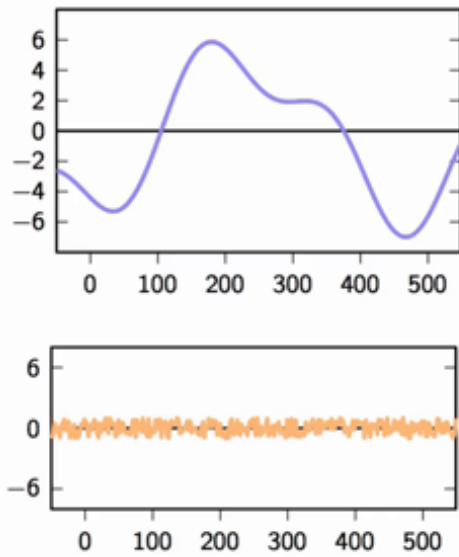
.....

.....

.....

.....

Typical filtering scenario: denoising



corrupted by some form of additive noise, which at this point we can just say it's some extraneous quantity that perturbs the measured value. So what you get in the end, instead of the smooth curve of the first panel, is a curve where you see that there's really fast wiggles that you interpret as being a noise corruption. And the question is, can we process this data as a discrete time signal so that most of the noise is removed and we get back a measurement that is in line with our expectations of smoothness? The first idea is to replace each sample of the input sequence with a local average.

notes

summary

0m 16s



Denoising by Moving Average

► idea: replace each sample by the local average

► for instance: $y[n] = (x[n] + x[n - 1])/2$

► more generally:

$$y[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n - k]$$

Varies are usually good when you want to eliminate random variations that you don't know much about. So for instance, assume a situation where we process the signal and the processed signal

notes

summary

0m 48s



Denoising by Moving Average

- ▶ idea: replace each sample by the local average
- ▶ for instance: $y[n] = (x[n] + x[n - 1])/2$
- ▶ more generally:

$$y[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n - k]$$

is at each sample the simple two-point average of the current sample and the past sample. Well, it could very well be that the two-point average is not enough to remove most of the noise. And so in general we could take longer averages by considering the output samples to be the average of capital M points from the input. If we try to do that experimentally, we can see immediately that the strategy works for

notes

summary

1m 2s



MA: impulse response

$$h[n] = \frac{1}{M} \sum_{k=0}^{M-1} \delta[n-k]$$
$$= \begin{cases} 1/M & \text{for } 0 \leq n < M \\ 0 & \text{otherwise} \end{cases}$$

increasing values of M . So if we take just a two-point average, we see that we don't really get a lot of leverage. You can see in blue the original signal and in red the filtered output. And with a two-point average we still have a lot of wiggles. But as we increase the number of points that we use to compute the average, we can see that we start to remove a little bit more of the noise. And by a 12-point average we already can see that the strategy actually pays off. We have already smoothed out the signal quite significantly. And if we're willing to increase the average up to say a hundred point, then we can see that the signal we get at the output is actually very smooth. You will also see that with respect to the original signal, the output has been delayed by a certain amount. This is the price we pay because we are actually using up a hundred samples to compute the output value. So in a sense we're accumulating a certain delay with respect to the input because we're averaging from the past. We will see this in more detail later on. What we have done is really a filtering operation. And if we want to compute the impulse response, we need to do nothing more than just put the delta function inside our averaging machine. And it's very easy to see that if we put the delta function in here, this is remember is the averaging relation that we saw before, we have that the impulse response is equal

notes

summary

1m 32s



MA: impulse response

$$h[n] = \frac{1}{M} \sum_{k=0}^{M-1} \delta[n-k]$$

$n=k$

$$= \begin{cases} 1/M & \text{for } 0 \leq n < M \\ 0 & \text{otherwise} \end{cases}$$

to 1 over m for n that goes from 0 to m minus 1 and 0 otherwise. It's easy to convince yourself that in this summation, there's only one non-zero term, which is the one where n is equal to k. Since the index goes from 0 to m minus 1, if your value of n is outside of this range, none of these terms will be non-zero and you will get 0. If we plot the impulse response, it looks like this.

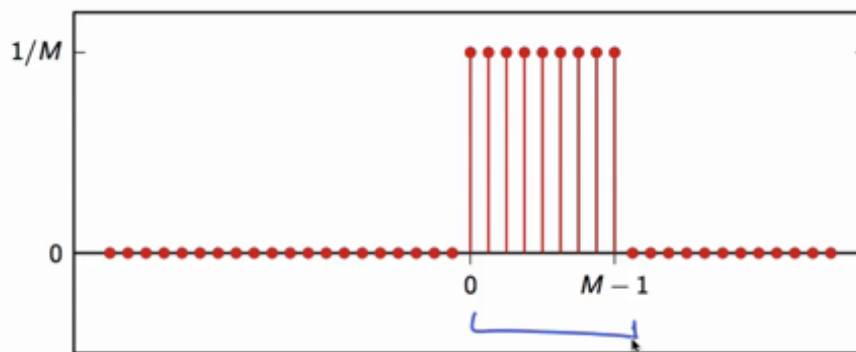
notes

summary

3m 7s



MA: impulse response



So you will have a string of samples that are equal to $1/M$ over m from 0 to m minus 1. So what we have learned experimentally is that the smoothing effect of the filter is

notes

summary

3m 34s



MA: analysis

- ▶ smoothing effect proportional to M
- ▶ number of operations and storage also proportional to M

proportional to the length of the impulse response. As a consequence of that, the number of operations that we have to perform to compute each output sample and also the storage, because we have to keep in the memory of our processing unit the past values in order to be able to compute the average, well both of these quantities are proportional to the length of the filter. So as the filter becomes longer and as its smoothing effect increases, the price we pay in terms of computational resources and memory are higher. So we might not like that and we might want to see whether there is a smarter way to achieve the same effect without using up so much in terms of computing power. Let's look at the equation for the moving average.

notes

summary

3m 45s



From the MA to a first-order recursion

$$y_M[n] = \frac{1}{M} (x[n] + x[n-1] + x[n-2] + \dots + x[n-M+1])$$

moving average over M points

It's just 1 over capital M times the sum of input samples from n to n minus capital M plus 1. Now we can split this into two parts. We have the first term, which is 1 over capital M times the current input plus 1 over capital

notes

summary

4m 34s



From the MA to a first-order recursion

$$y_M[n] = \underbrace{\frac{1}{M}x[n]}_{\text{current sample}} + \frac{1}{M}(x[n-1] + x[n-2] + \dots + x[n-M+1])$$

M times the sum of past input samples from n minus 1 to n minus capital M plus 1.

notes

summary

4m 47s



From the MA to a first-order recursion

$$y_M[n] = \underbrace{\frac{1}{M}}_{\text{}} x[n] + \frac{1}{M} \underbrace{(x[n-1] + x[n-2] + \dots + x[n-M+1])}_{\text{}} \quad \text{with a blue arc above the sum term}$$

And this is the sum of capital N minus 1 terms.

notes

summary

4m 49s



From the MA to a first-order recursion

$$y_M[n] = \underbrace{\frac{1}{M}}_{\text{1}} x[n] + \frac{1}{M} \underbrace{(x[n-1] + x[n-2] + \dots + x[n-M+1])}_{\text{M-1}}$$

So the second term here looks suspiciously close to the moving average computed this

notes

summary

5m 1s



From the MA to a first-order recursion

$$y_M[n] = \frac{1}{M}x[n] + \frac{1}{M}(x[n-1] + x[n-2] + \dots + x[n-M+1])$$

"almost" $y_{M-1}[n-1]$

i.e., moving average over $M-1$ points, delayed by one

time over capital M minus 1 points, so 1 point less than before, and delayed by 1. So it's close but it's not quite the same thing. So let's look at ways to formalize this relationship. To do so we first write out the standard equation for the moving average filter, which we have

notes

summary

5m 4s



From the MA to a first-order recursion

Formally:

$$y_M[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n-k]$$

$$y_M[n-1] = \frac{1}{M} \sum_{k=0}^{M-1} x[(n-1)-k]$$

seen before so nothing new here. Now we try and compute the delayed output. So y of capital M of n minus 1 is the sum, it's 1 over capital M times the sum from k that goes from 0 to capital M minus 1 of x of n minus 1 minus k . We can rearrange the index of the signal like so.

notes

summary

5m 25s



From the MA to a first-order recursion

Formally:

$$y_M[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n-k]$$

$$y_M[n-1] = \frac{1}{M} \sum_{k=1}^M x[n-k]$$

And now this plus 1 that affects only the summation index can be propagated to the summation range and we have that if we delay the moving average by 1 we basically just affect the

notes

summary

5m 49s



From the MA to a first-order recursion

Formally:

$$y_M[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n-k]$$

$$y_M[n-1] = \frac{1}{M} \sum_{k=1}^M x[n-k]$$

range of the summation that now goes from 1 to M instead of from 0 to M minus 1. Okay, now let's look at the formula for the moving average over M minus 1 point.

notes

summary

6m 1s



From the MA to a first-order recursion

Formally:

$$y_M[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n-k]$$

$$y_M[n-1] = \frac{1}{M} \sum_{k=1}^M x[n-k]$$

$$y_{M-1}[n] = \frac{1}{M-1} \sum_{k=0}^{M-2} x[n-k]$$

And this is just the moving average, so 1 over capital M minus 1 times the sum from k that goes to 0 to capital M minus 2 of n of x minus k. Now we do the same, we try to delay this by 1 and again this delay will propagate to the

notes

summary

6m 12s



From the MA to a first-order recursion

Formally:

$$y_M[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n-k]$$

$$y_M[n-1] = \frac{1}{M} \sum_{k=1}^M x[n-k]$$

$$y_{M-1}[n-1] = \frac{1}{M-1} \sum_{k=1}^{M-1} x[n-k]$$

summation range and we will have the sum that goes now from k is equal to 1 to capital M

notes

summary

6m 31s



From the MA to a first-order recursion

$$y_M[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n-k] \quad y_{M-1}[n-1] = \frac{1}{M-1} \sum_{k=1}^{M-1} x[n-k]$$

$$\sum_{k=0}^{M-1} x[n-k] = x[n] + \sum_{k=1}^{M-1} x[n-k]$$

$$My_M[n] = x[n] + (M-1)y_{M-1}[n-1]$$

minus 1. So formula for the moving average over capital M points and here formula for the moving average

notes

summary

6m 41s



From the MA to a first-order recursion

$$y_M[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n-k] \quad y_{M-1}[n-1] = \frac{1}{M-1} \sum_{k=1}^{M-1} x[n-k]$$

$$\underbrace{\sum_{k=0}^{M-1} x[n-k]}_{\text{blue bracket}} = x[n] + \sum_{k=1}^{M-1} x[n-k]$$

$$My_M[n] = x[n] + (M-1)y_{M-1}[n-1]$$

over $M-1$ points delayed by 1. Alright, but now if we take the sum of the input from 0 to capital $M-1$ we can split this as current sample plus $M-2$ samples in the past.

notes

summary

6m 49s



From the MA to a first-order recursion

$$\begin{aligned}
 y_M[n] &= \frac{1}{M} \sum_{k=0}^{M-1} x[n-k] & y_{M-1}[n-1] &= \frac{1}{M-1} \sum_{k=1}^{M-1} x[n-k] \\
 &\downarrow & & \swarrow \quad \searrow \\
 \sum_{k=0}^{M-1} x[n-k] &= x[n] + \sum_{k=1}^{M-1} x[n-k] \\
 My_M[n] &= x[n] + (M-1)y_{M-1}[n-1]
 \end{aligned}$$

And so this part here corresponds to this and this part here corresponds to this, right?
 So we can swap terms and we obtain the relationship that we were looking for between the moving

notes

summary

7m 4s



From the MA to a first-order recursion

$$y_M[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n-k] \quad y_{M-1}[n-1] = \frac{1}{M-1} \sum_{k=1}^{M-1} x[n-k]$$

$$\sum_{k=0}^{M-1} x[n-k] = x[n] + \sum_{k=1}^{M-1} x[n-k]$$

$$\underline{M y_M[n]} = x[n] + (M-1) \underline{y_{M-1}[n-1]}$$

average over capital M points and the moving average over capital M minus 1 points delayed by 1. So we rearrange the terms and we have that the moving average over capital M points in

notes

summary

7m 18s



From the MA to a first-order recursion

$$y_M[n] = \frac{M-1}{M}y_{M-1}[n-1] + \frac{1}{M}x[n]$$

$$y_M[n] = \lambda y_{M-1}[n-1] + (1-\lambda)x[n], \quad \lambda = \frac{M-1}{M}$$

n is equal to capital M minus 1 over n times the moving average over M minus 1 points delayed

notes

summary

7m 35s



From the MA to a first-order recursion

$$y_M[n] = \frac{M-1}{M}y_{M-1}[n-1] + \frac{1}{M}x[n]$$

$$y_M[n] = \lambda y_{M-1}[n-1] + (1-\lambda)x[n], \quad \lambda = \frac{M-1}{M}$$

by 1 plus 1 over m times x of n. And so if we want to write this a little bit better we define a parameter lambda as M minus 1 over m and we have this finite relationship here.

notes

summary

7m 39s

