



Course material

Course:

Digital Signal Processing II

Video:

2.1.2.b The leaky integrator

Concepts (extracted from automatically generated subtitles):

Output values. Lambda times. Minus lambda times. Next step. Previous equation. Previous output values. Leakey integrator filter. Impulse response. Lambda. Delta function. Values of lambda. Current value of the output. Value of lambda. Minus infinity. M points.



[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/>

From the MA to a first-order recursion

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

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

Now, there's nothing new what we saw so far. We just rearranged terms in finite sums. But here comes the trick that defines the Leakey integrator filter. When m is very large, we assume that the average over $m-1$ or over m points is going to be pretty much the same.

notes

summary

0m 0s



The Leaky Integrator

► when M is large, $y_{M-1}[n] \approx y_M[n]$ (and $\lambda \approx 1$)

► try the filter

$$y[n] = \lambda y[n-1] + (1 - \lambda)x[n]$$

► filter is now recursive, since it uses its previous output value

Not much is going to change if we add a single data point to a very long average. And also, when m is very large, λ , which was defined as $m-1$ over m , is going to be very close to 1.

notes

summary

0m 9s



The Leaky Integrator

► when M is large, $y_{M-1}[n] \approx y_M[n]$ (and $\lambda \approx 1$)

► try the filter

$$y[n] = \lambda y[n-1] + (1 - \lambda)x[n]$$

► filter is now recursive, since it uses its previous output value

So, we can rewrite the previous equation as current estimate for the moving average is equal to lambda times the previous estimate for a moving average,

notes

summary

0m 24s



The Leaky Integrator

► when M is large, $y_{M-1}[n] \approx y_M[n]$ (and $\lambda \approx 1$)

► try the filter

$$y[n] = \lambda y[n-1] + (1 - \lambda)x[n]$$

► filter is now recursive, since it uses its previous output value

plus 1 minus lambda times the current input.

notes

summary

0m 25s



The Leaky Integrator

► when M is large, $y_{M-1}[n] \approx y_M[n]$ (and $\lambda \approx 1$)

► try the filter

$$y[n] = \lambda y[n-1] + (1 - \lambda)x[n]$$



► filter is now recursive, since it uses its previous output value

And this defines a filter that we can use to process our signal. Now, the filter is recursive because we're using output values, previous output values, to compute the current value of the output. So, there's a feedback path that goes from the output back into the filter.

notes

summary

0m 37s



The Leaky Integrator

► when M is large, $y_{M-1}[n] \approx y_M[n]$ (and $\lambda \approx 1$)

► try the filter

$$y[n] = \lambda y[n-1] + (1 - \lambda)x[n]$$



► filter is now recursive, since it uses its previous output value

Before we even try to analyze this, let's see if it works.

notes

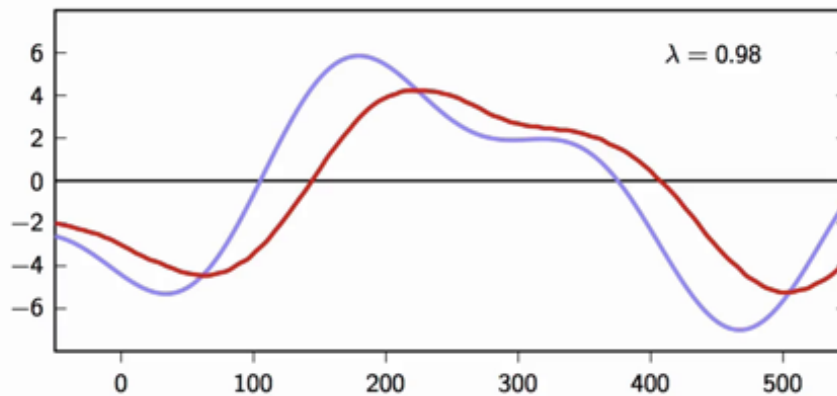
summary

0m 58s



Denoising recursively with the Leaky Integrator

$$y[n] = \lambda y[n-1] + (1-\lambda)x[n]$$



So, we take our smooth signal corrupted by noise and we filter it with a Leaky integrator for varying values of lambda. So, if we start with lambda rather small, we don't really see much happening. But as we increase the value of lambda towards 1, which is really the operating assumption for the derivation of the filter anyway, we see that we start to smooth the signal as we expected. And when lambda is very close to 1, the smoothing power is comparable to that of a moving average filter with a very high number of taps. But now the difference is that each output value only requires three operations, because remember, the value is simply one multiplication and then there's going to be another multiplication and one sum.

notes

summary

1m 1s



What about the impulse response?

$$y[n] = \lambda y[n-1] + (1-\lambda)\delta[n]$$

- ▶ $y[n] = 0$ for all $n < 0$
- ▶ $y[0] = \lambda y[-1] + (1-\lambda)\delta[0] = (1-\lambda)$
- ▶ $y[1] = \lambda y[0] + (1-\lambda)\delta[1] = \lambda(1-\lambda)$
- ▶ $y[2] = \lambda y[1] + (1-\lambda)\delta[2] = \lambda^2(1-\lambda)$
- ▶ $y[3] = \lambda y[2] + (1-\lambda)\delta[3] = \lambda^3(1-\lambda)$
- ▶ ...

And this is independent of the value of lambda, so high smoothing power at a fixed price. Of course, the natural question when we have a filter is what the impulse response is going to be.

notes

summary

1m 49s



What about the impulse response?

$$y[n] = \lambda y[n-1] + (1-\lambda)\delta[n]$$

- ▶ $y[n] = 0$ for all $n < 0$
- ▶ $y[0] = \lambda y[-1] + (1-\lambda)\delta[0] = (1-\lambda)$
- ▶ $y[1] = \lambda y[0] + (1-\lambda)\delta[1] = \lambda(1-\lambda)$
- ▶ $y[2] = \lambda y[1] + (1-\lambda)\delta[2] = \lambda^2(1-\lambda)$
- ▶ $y[3] = \lambda y[2] + (1-\lambda)\delta[3] = \lambda^3(1-\lambda)$
- ▶ ...

And to compute the impulse response of the Leakey integrator, all we need to do is to plug the delta function into the recursive equation and see what happens. So, when n is less than 0, since the delta function is 0 from minus infinity to 0, nothing non-zero has ever happened inside this equation. And so it's very safe to assume that the output will be 0 for all values of n less than 0.

notes

summary

2m 1s



What about the impulse response?

$$y[n] = \lambda y[n-1] + (1-\lambda)\delta[n]$$

- ▶ $y[n] = 0$ for all $n < 0$
- ▶ $y[0] = \lambda y[-1] + (1-\lambda)\delta[0] = (1-\lambda)$
- ▶ $y[1] = \lambda y[0] + (1-\lambda)\delta[1] = \lambda(1-\lambda)$
- ▶ $y[2] = \lambda y[1] + (1-\lambda)\delta[2] = \lambda^2(1-\lambda)$
- ▶ $y[3] = \lambda y[2] + (1-\lambda)\delta[3] = \lambda^3(1-\lambda)$
- ▶ ...

Things start to happen when n reaches 0, at which point the delta function kicks and assumes the value of 1.

notes

summary

2m 24s



What about the impulse response?

$$y[n] = \lambda y[n-1] + (1 - \lambda)\delta[n]$$

- ▶ $y[n] = 0$ for all $n < 0$
- ▶ $y[0] = \lambda y[-1] + (1 - \lambda)\delta[0] = (1 - \lambda)$
- ▶ $y[1] = \lambda y[0] + (1 - \lambda)\delta[1] = \lambda(1 - \lambda)$
- ▶ $y[2] = \lambda y[1] + (1 - \lambda)\delta[2] = \lambda^2(1 - \lambda)$
- ▶ $y[3] = \lambda y[2] + (1 - \lambda)\delta[3] = \lambda^3(1 - \lambda)$
- ▶ ...

So if we compute the value of the output for n equal to 0, we have λ times the previous value of the output, which we know to be 0, so this is 0, plus 1 minus λ times the value of δ and 0, which is 1. And so the output will be 1 minus λ . At the next step, y of 1 is equal to λ times the previous value of the output, which now is not non-zero, is 1 minus λ , as we computed before,

notes

summary

2m 25s



What about the impulse response?

$$y[n] = \lambda y[n-1] + (1 - \lambda)\delta[n]$$

- ▶ $y[n] = 0$ for all $n < 0$
- ▶ $y[0] = \lambda y[-1] + (1 - \lambda)\delta[0] = (1 - \lambda)$
- ▶ $y[1] = \lambda y[0] + (1 - \lambda)\delta[1] = \lambda(1 - \lambda)$
- ▶ $y[2] = \lambda y[1] + (1 - \lambda)\delta[2] = \lambda^2(1 - \lambda)$
- ▶ $y[3] = \lambda y[2] + (1 - \lambda)\delta[3] = \lambda^3(1 - \lambda)$
- ▶ ...

plus 1 minus lambda times delta of 1, which we know to be 0, and so this is cancelled. And the final output is lambda times 1 minus lambda. At the next step, once again the recursive equation kicks in, and we have lambda times the previous value,

notes

summary

3m 1s



What about the impulse response?

$$y[n] = \lambda y[n-1] + (1 - \lambda)\delta[n]$$

- ▶ $y[n] = 0$ for all $n < 0$
- ▶ $y[0] = \lambda y[-1] + (1 - \lambda)\delta[0] = (1 - \lambda)$
- ▶ $y[1] = \lambda y[0] + (1 - \lambda)\delta[1] = \lambda(1 - \lambda)$
- ▶ $y[2] = \lambda y[1] + (1 - \lambda)\delta[2] = \lambda^2(1 - \lambda)$
- ▶ $y[3] = \lambda y[2] + (1 - \lambda)\delta[3] = \lambda^3(1 - \lambda)$
- ▶ ...

which we know to be λ times $1 - \lambda$, plus $1 - \lambda$ delta of 2, but again delta from now on is going to be 0, so this term we don't even have to look at. And so here we have λ^2 times $1 - \lambda$. The next step is going to be the same, and you have pretty much understood what's going on here. If we plot the impulse response, we have an exponentially decaying function,

notes

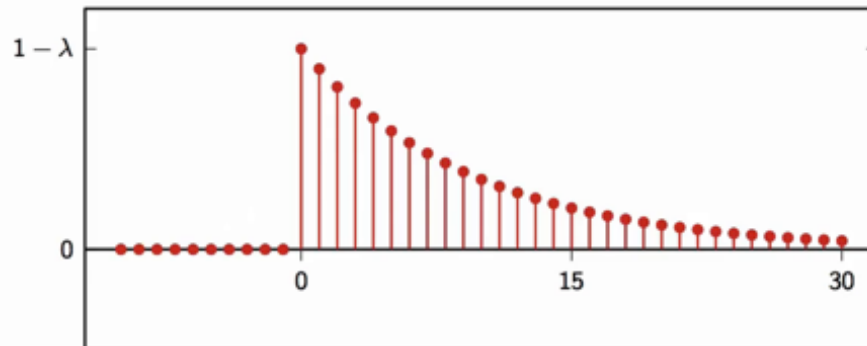
summary

3m 13s



Impulse response

$$h[n] = (1 - \lambda)\lambda^n u[n]$$



where the exponential basis is lambda, and the whole curve is weighted by 1 minus lambda,

notes

summary

3m 40s



Leaky Integrator: why the name

Discrete-time integrator is a boundless accumulator:

$$y[n] = \sum_{k=-\infty}^n x[k]$$

We can rewrite the integrator as

$$y[n] = y[n-1] + x[n]$$

and it's of course multiplied by a unistep, because everything is 0 for n less than 0. You might wonder why we call this filter a leaky integrator. The reason is because if you consider for instance this very simple sum, it's the sum of all input samples from minus infinity to current time. This is really the discrete time approximation to an integral, because you're really accumulating all past values up to now. And we can rewrite this summation here as such. So at each step we get what we have accumulated up to the previous step, and we sum the current input.

notes

summary

3m 49s



Leaky Integrator: why the name

To prevent "explosion" pick $\lambda < 1$

$$y[n] = \lambda y[n-1] + (1 - \lambda)x[n]$$

keep only a fraction λ of
the accumulated value
so far and forget
("leak") a fraction $1 - \lambda$

add only a fraction $1 - \lambda$
of the current value to
the accumulator

So you see it's very close to the equation for the leaky integrator, except in the leaky integrator what we do is the following. We scale the previous accumulated value by lambda, which means we...

notes

summary

4m 25s



Leaky Integrator: why the name

To prevent "explosion" pick $\lambda < 1$

$$y[n] = \lambda y[n-1] + (1 - \lambda)x[n]$$

keep only a fraction λ of
the accumulated value
so far and forget
("leak") a fraction $1 - \lambda$

add only a fraction $1 - \lambda$
of the current value to
the accumulator

Remember lambda is close to 1, so that means we keep pretty much all of what we have accumulated before,

notes

summary

4m 37s



Leaky Integrator: why the name

To prevent "explosion" pick $\lambda < 1$

$$y[n] = \lambda y[n-1] + (1 - \lambda)x[n]$$

keep only a fraction λ of
the accumulated value
so far and forget
("leak") a fraction $1 - \lambda$

add only a fraction $1 - \lambda$
of the current value to
the accumulator

but we forget a little bit. We leak a part of it. If lambda is equal to 0.9, then it means that 10% of what we have accumulated is lost. And we replace what we've lost by a fraction of the input, and this fraction is 1 minus lambda.

notes

summary

4m 39s



Leaky Integrator: why the name

To prevent "explosion" pick $\lambda < 1$

$$y[n] = \lambda y[n-1] + (1 - \lambda)x[n]$$

keep only a fraction λ of
the accumulated value
so far and forget
("leak") a fraction $1 - \lambda$

add only a fraction $1 - \lambda$
of the current value to
the accumulator

So what we've lost from the accumulator we replace with an equal percentage of the input.

notes

summary

5m 0s



Leaky Integrator: why the name

To prevent "explosion" pick $\lambda < 1$

$$y[n] = \lambda y[n-1] + (1 - \lambda)x[n]$$

keep only a fraction λ of
the accumulated value
so far and forget
("leak") a fraction $1 - \lambda$

add only a fraction $1 - \lambda$
of the current value to
the accumulator

The idea is that by picking lambda less than 1, we are forgetting, we're leaking the past, and the accumulation will never blow up.

notes

summary

5m 1s

