Filtering and denoising

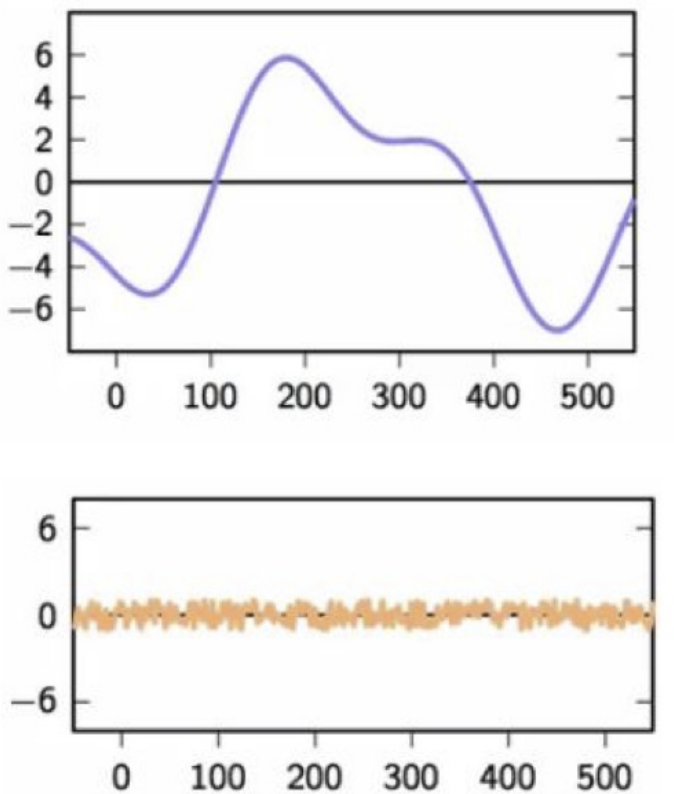
Chiara Ravazzi

National Research Council of Italy
Institute of Electronics, Computer and Telecommunication Engineering, c/o Politecnico di Torino, Italy

Systems Modeling & Control Group

Analisi tempo-frequenza e multiscala – 01RMQNG

Typical filtering scenario : denoising

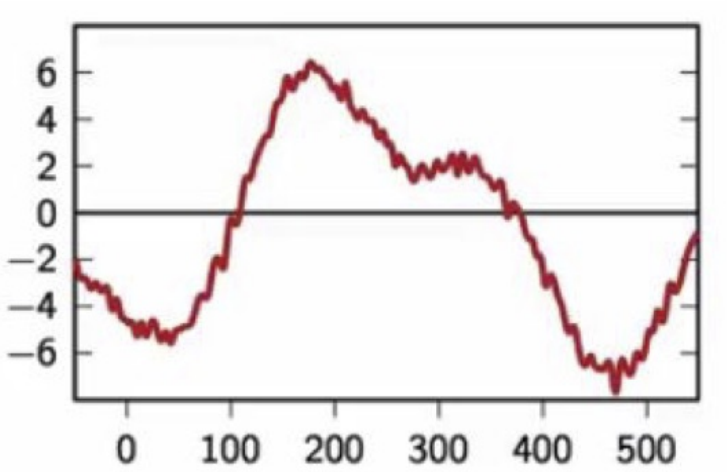


Denoising

--> Alcune misure sono presentano rumore, ad esempio rumore bianco (rumore gaussiano con media nulla)

--> Il segnale che abbiamo è formato dal segnale originale + il rumore bianco

--> Il processo di denoising è quello di cercare di ottenrre il segnalee originale senza più avere il rumore sovrapposto



Denoising by Moving Average

- idea : replace each sample by the local average
- example : $y_n = (x_n + x_{n-1})/2$
- more generally :

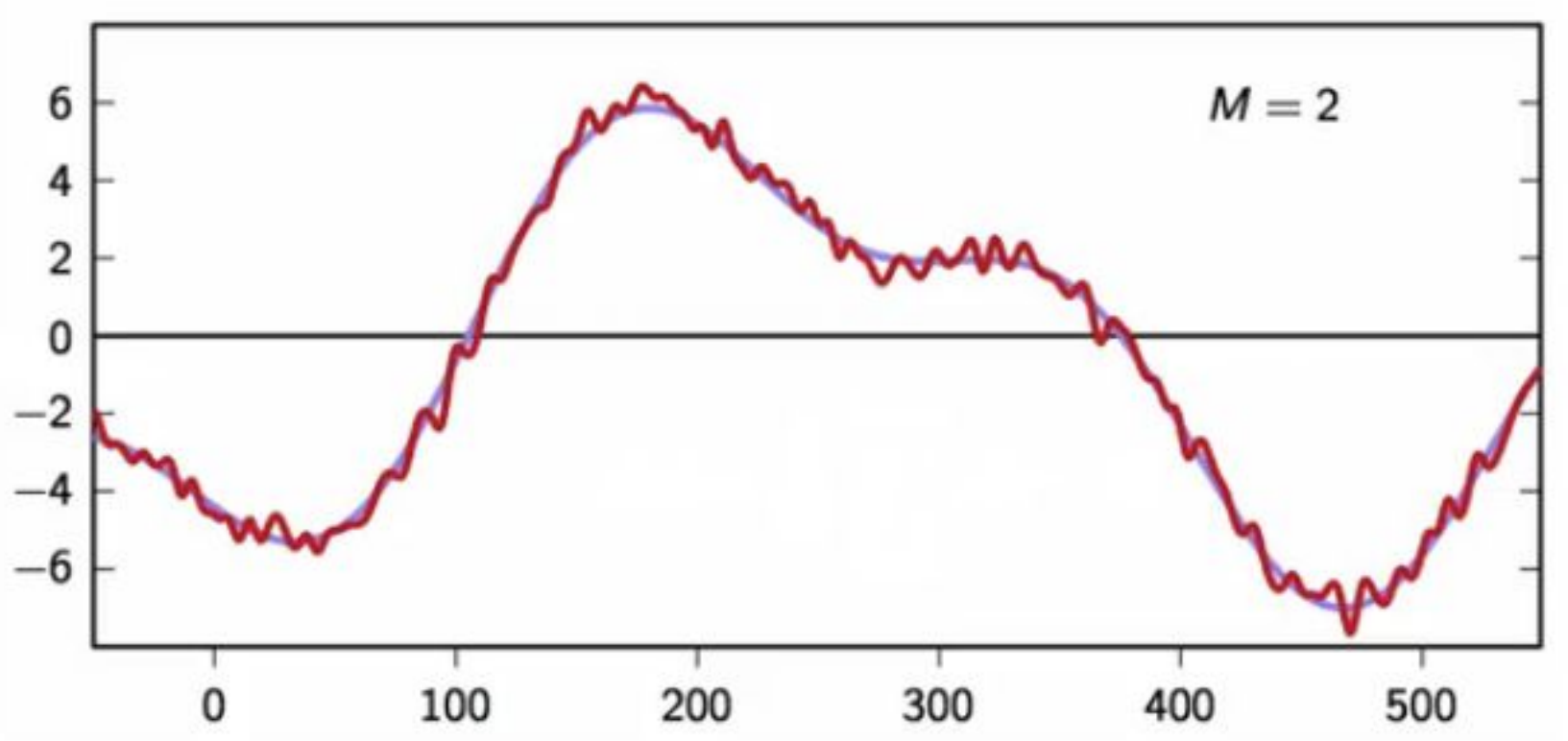
$$y_n = \frac{1}{M} \sum_{k=0}^{M-1} x_{n-k}$$

Denoising by Moving Average

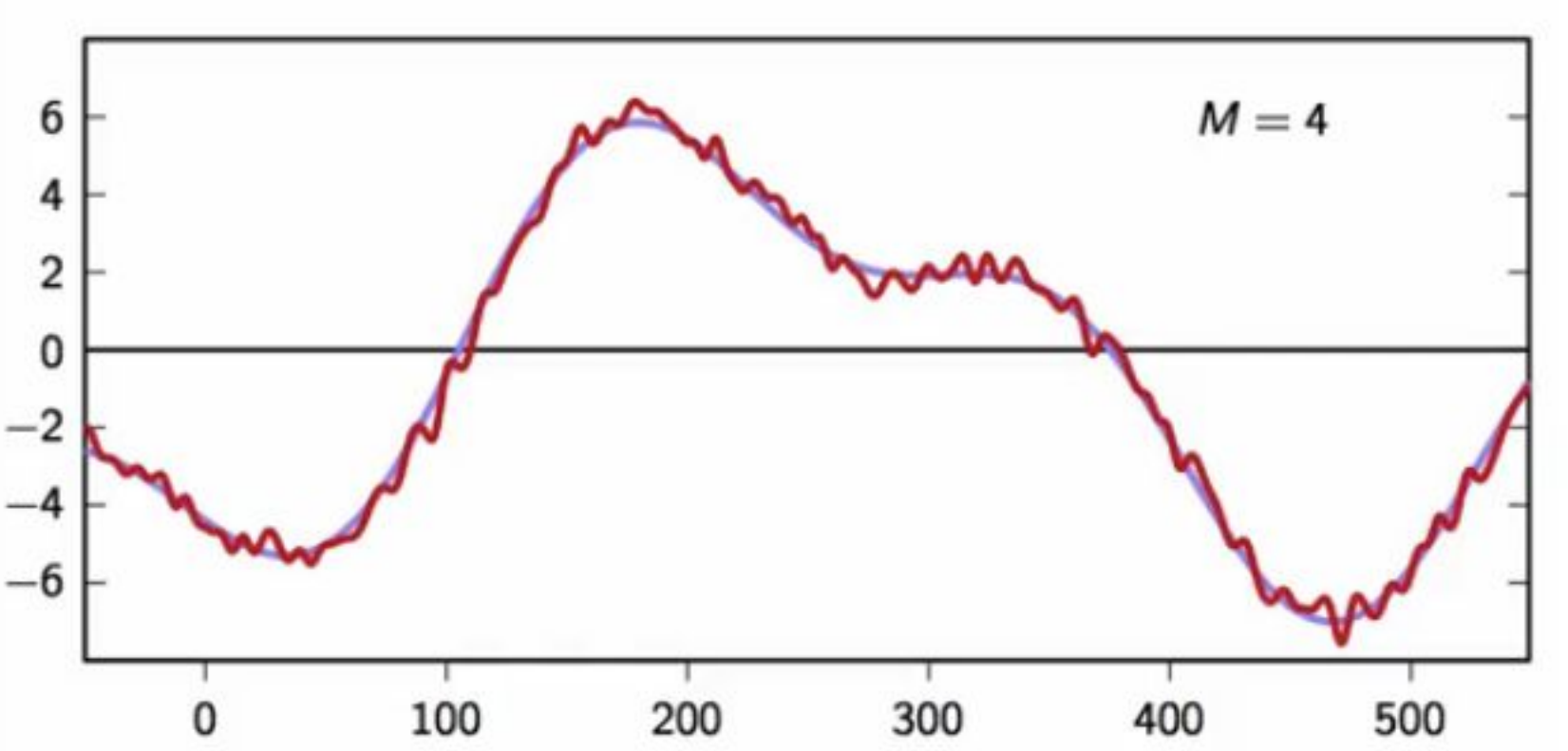
M == finestra temporale

Aumentando la frequenza, si ottengono oscillazioni sempre migliori e più smooth (regolarizzazione maggiore), ma il segnale, se aumentiamo molto M, si allungano sempre di più

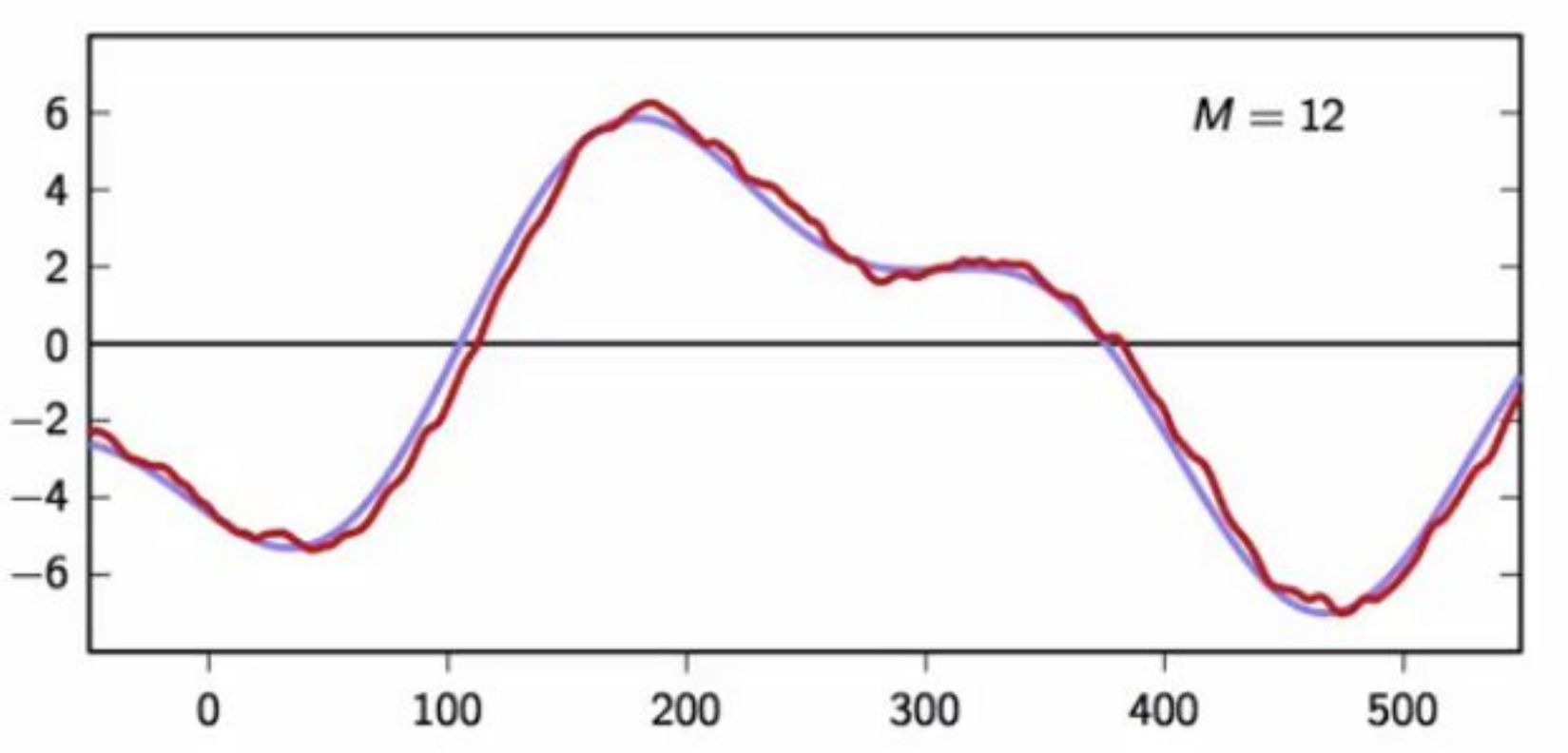
--> bisogna trovare un trade off



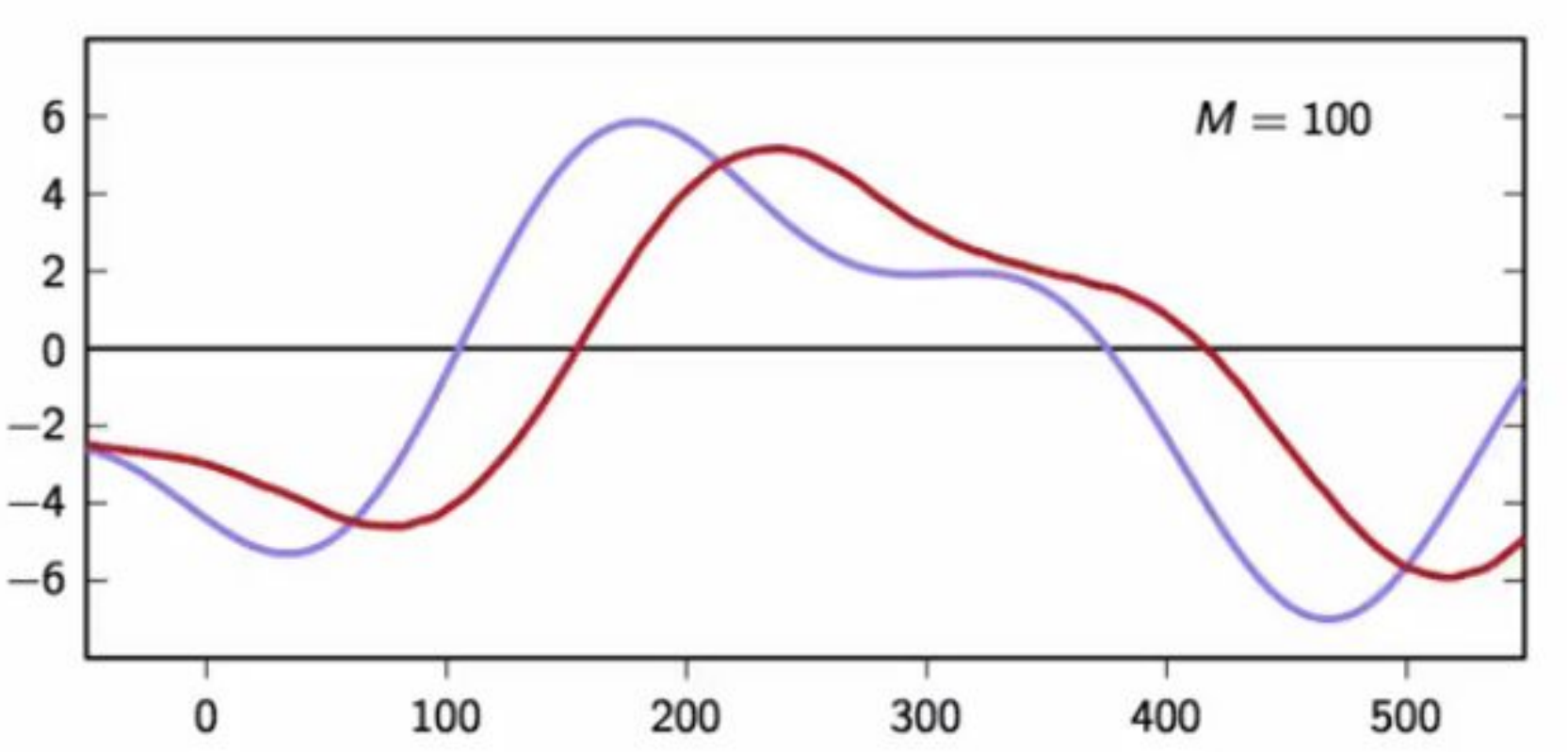
Denoising by Moving Average



Denoising by Moving Average

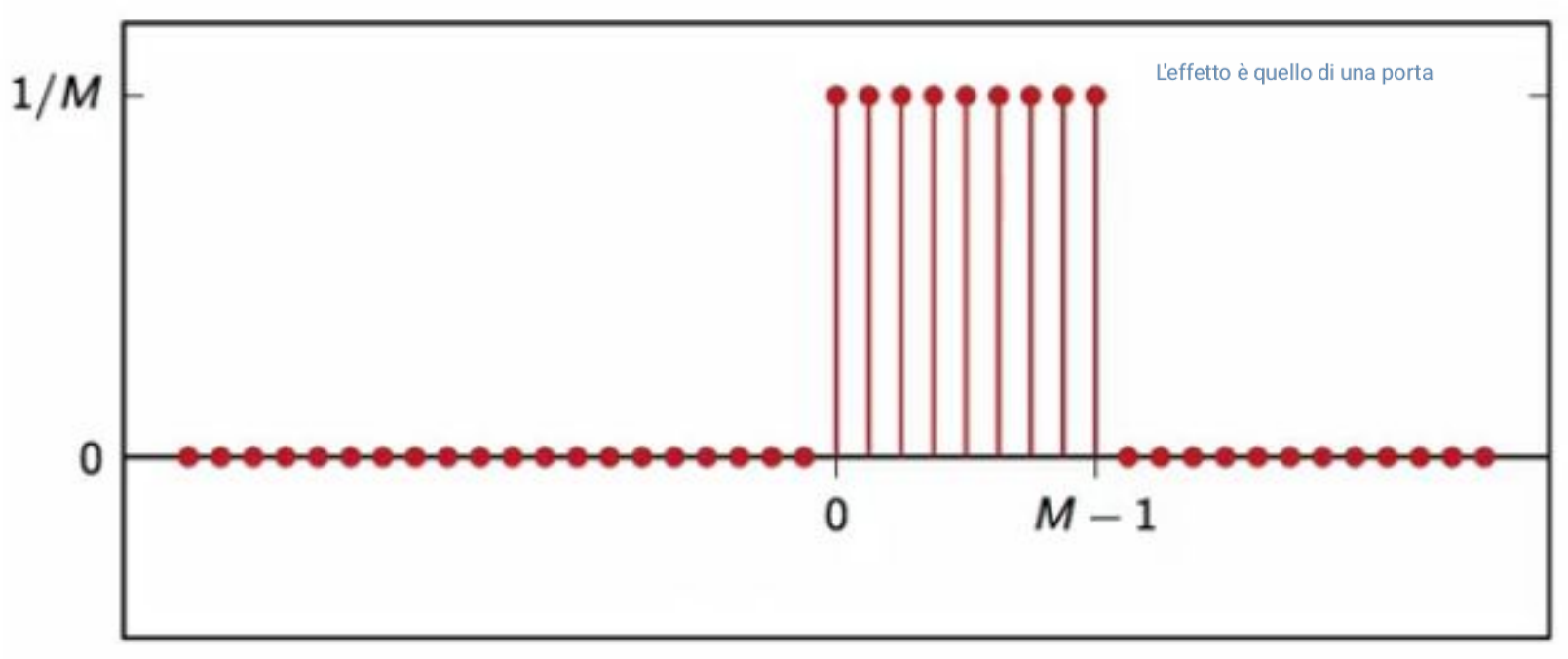


Denoising by Moving Average



Moving average filter : impulse response

$$h_n = \frac{1}{M} \sum_{k=0}^{M-1} \delta_{n-k} = \begin{cases} \frac{1}{M} & \text{for } 0 \leq n < M \\ 0 & \text{otherwise} \end{cases}$$



From the moving average to a first-order recursion

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

moving average over M points

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

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

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

From the moving average to a first-order recursion

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

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

L'uscita del sistema è una combinazione convessa di quelli precedenti

A seconda del valore di lambda il risultato si avvicina maggiormente o meno all'uscita precendete

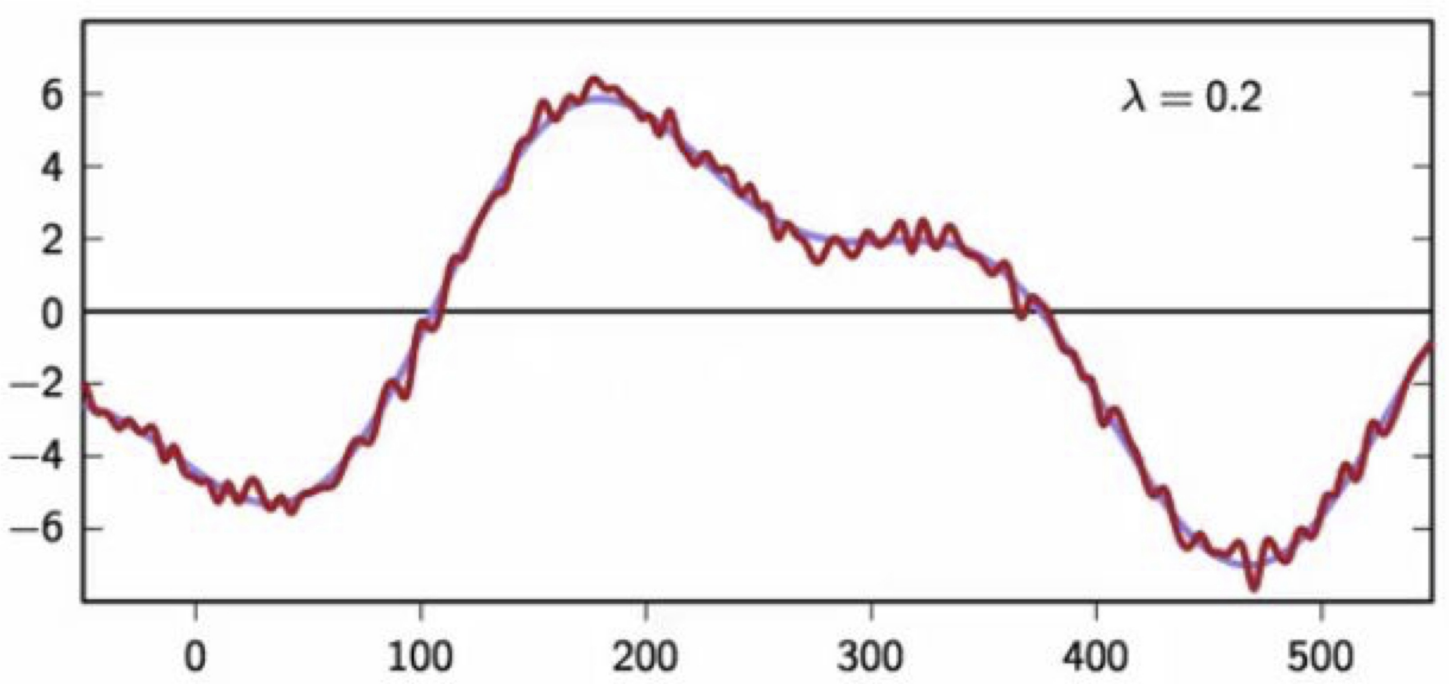
Leaky integrator

- when M is large, $y_M[N] \approx y_{M-1}[N]$ (and $\lambda \approx 1$)
- Leaky integrator:

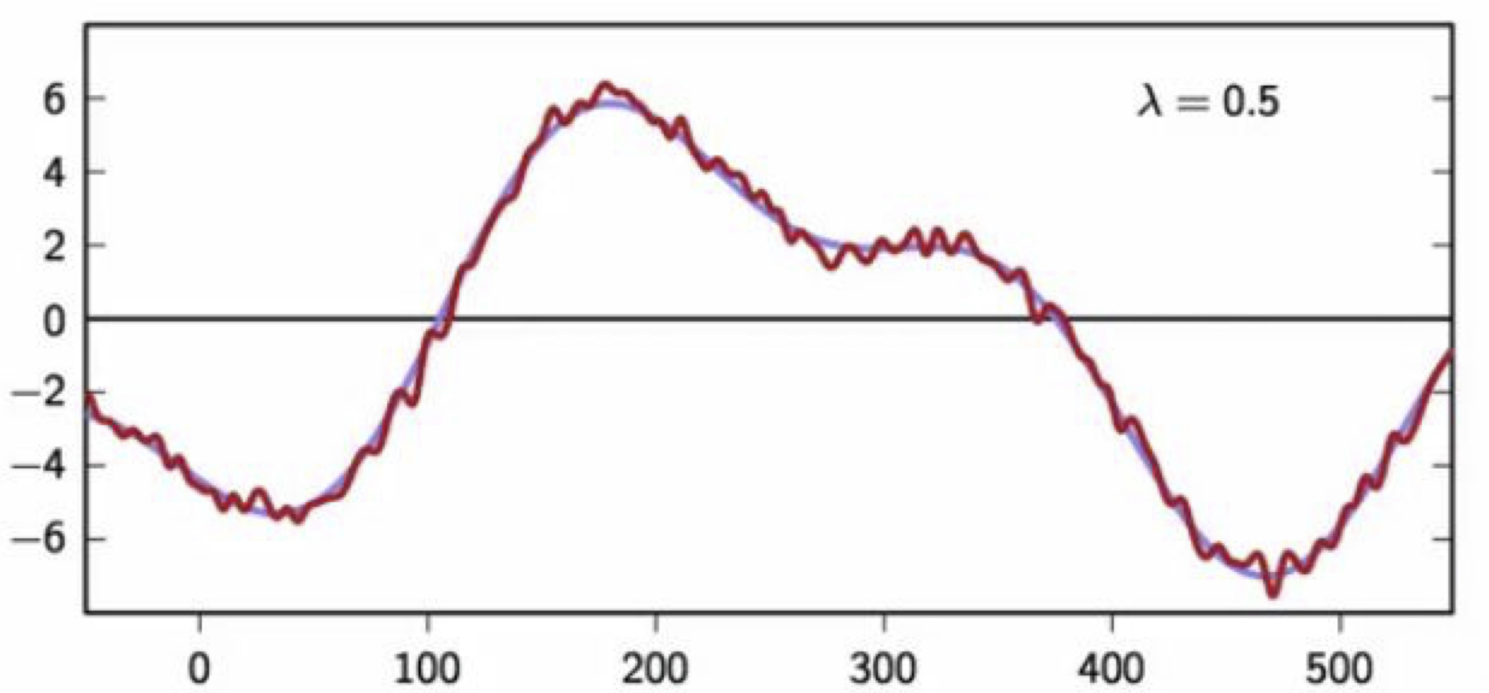
$$y_n = \lambda y_{n-1} + (1 - \lambda) x_n$$

- the filter is now recursive, since it uses it previous output value

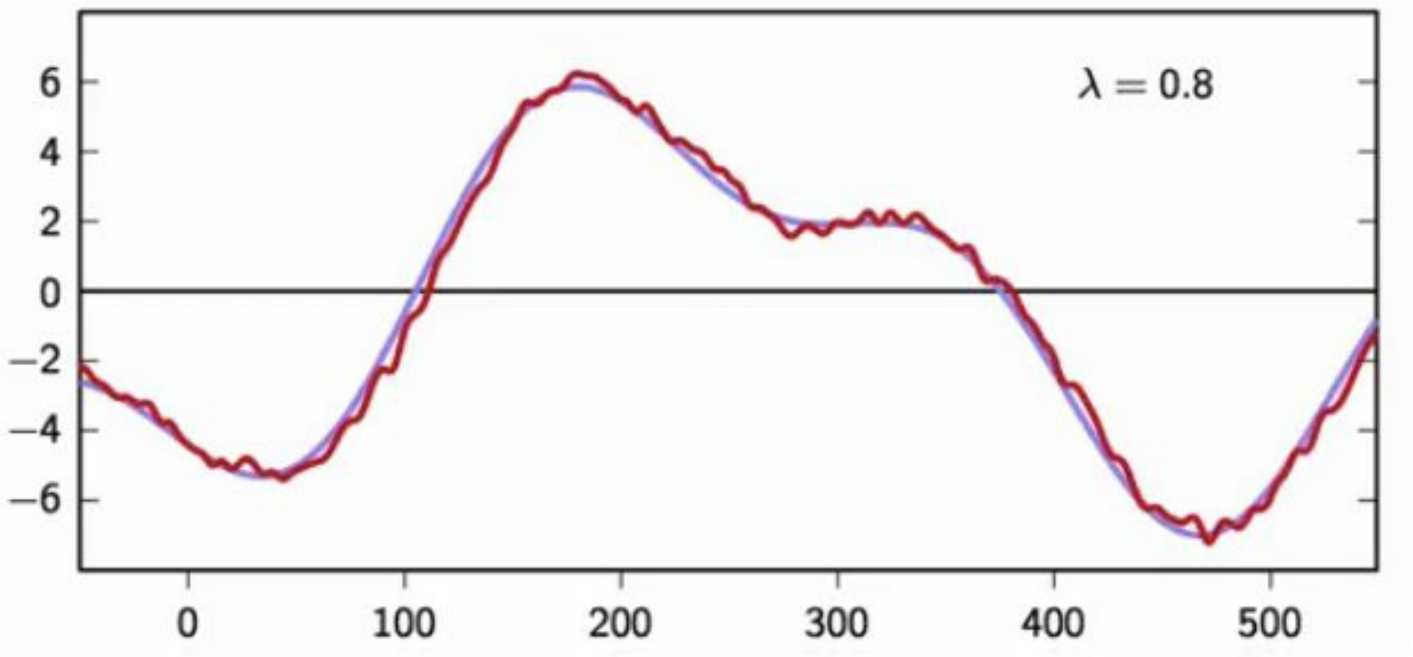
From the moving average to a first-order recursion



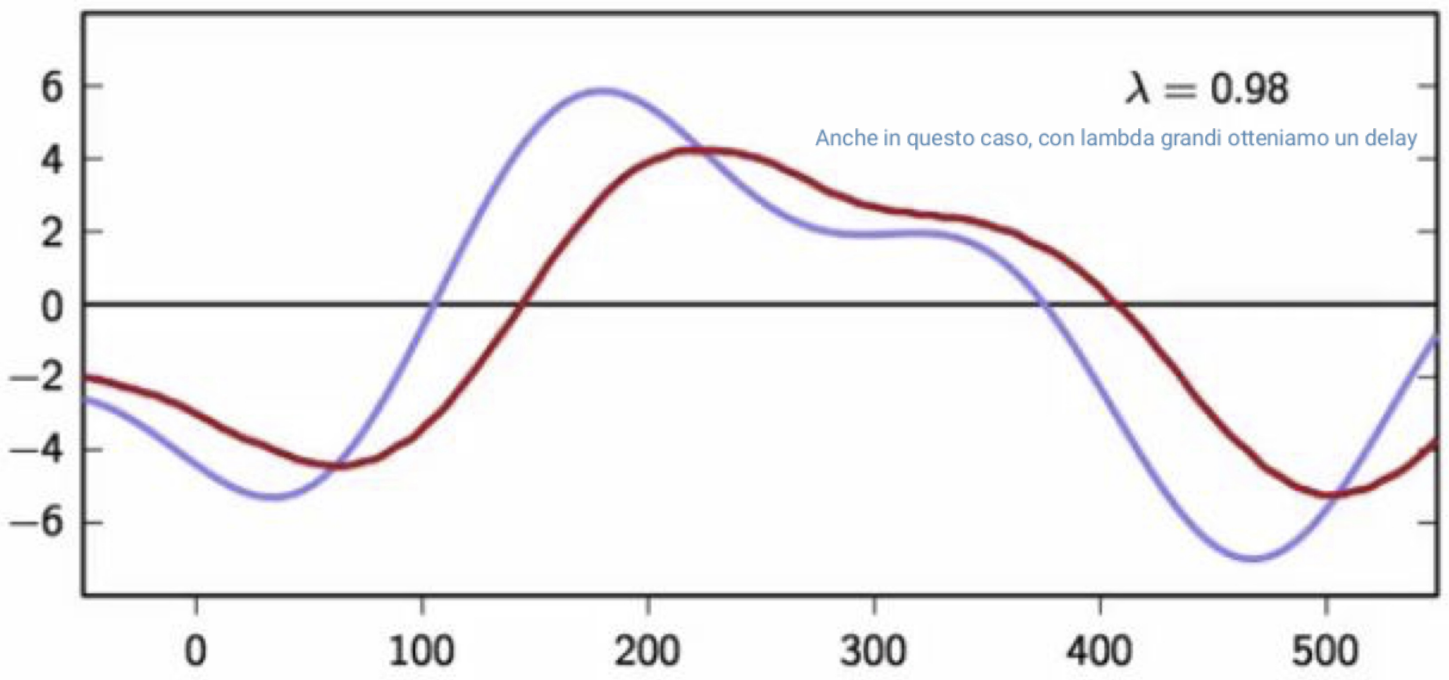
From the moving average to a first-order recursion



From the moving average to a first-order recursion



From the moving average to a first-order recursion



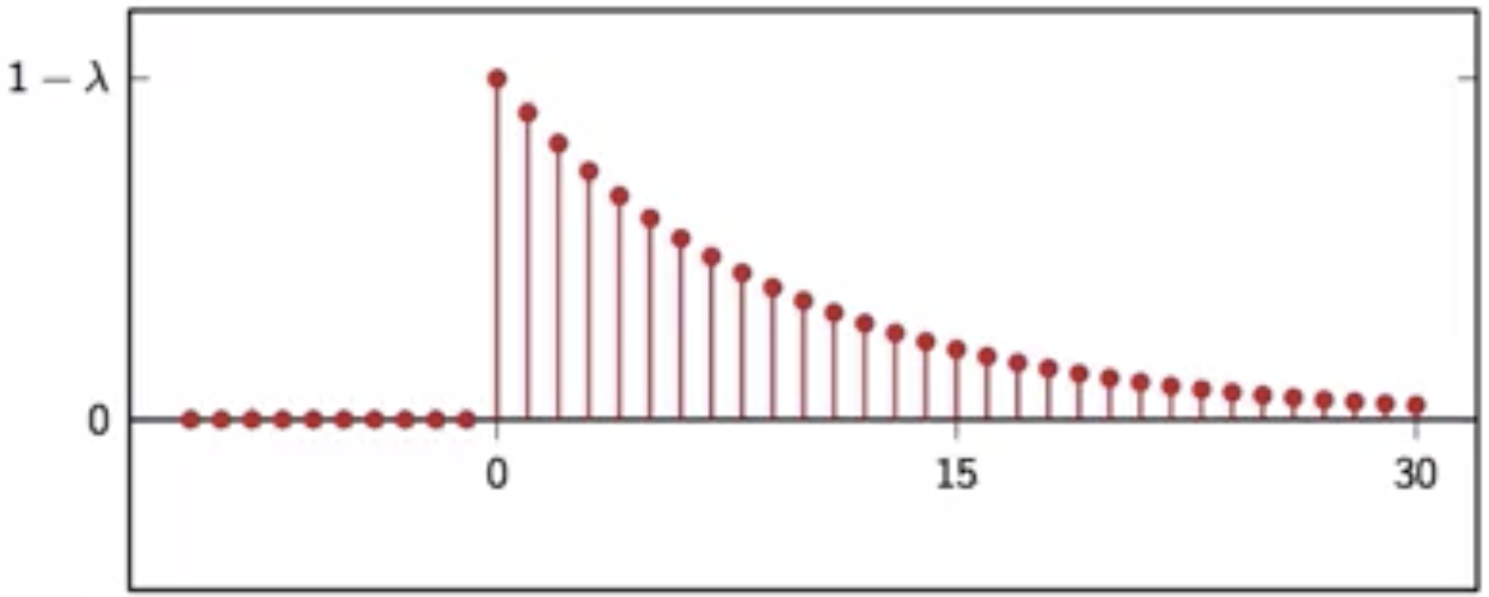
Impulse response

$$h_n = \lambda h_{n-1} + (1-\lambda)\delta_n$$

- $h_n = 0, \quad \forall n < 0$
- $h_0 = \lambda h_{-1} + (1-\lambda)\delta_0 = (1-\lambda)$
- $h_1 = \lambda h_0 + (1-\lambda)\delta_1 = \lambda(1-\lambda)$
- $h_2 = \lambda h_1 + (1-\lambda)\delta_2 = \lambda^2(1-\lambda)$
- $h_3 = \lambda h_2 + (1-\lambda)\delta_3 = \lambda^3(1-\lambda)$
- $\ldots$

From the moving average to a first-order recursion

$$h_n = \lambda^n (1 - \lambda) u_n$$



Leaky integrator

Discrete-time integrator is a boundless accumulator

$$y_n = \sum_{k=-\infty}^{n} x_k$$

or, equivalently,

$$y_n = y_{n-1} + x_n$$

Leaky integrator

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

$$y_n = \lambda y_{n-1} + (1 - \lambda) x_n$$

I campioni più lontani tendono ad avere meno importanza rispetto a quelli più vicini

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