

# Digital Signal Processing

Module 2: Discrete-time signals

Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

# Video Introduction

Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

- ▶ **Module 2.1:** discrete-time signals and operators
- ▶ **Module 2.2:** the discrete-time complex exponential
- ▶ **Module 2.3:** the Karplus-Strong algorithm

# Digital Signal Processing

Module 2.1: Discrete-time signals

Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

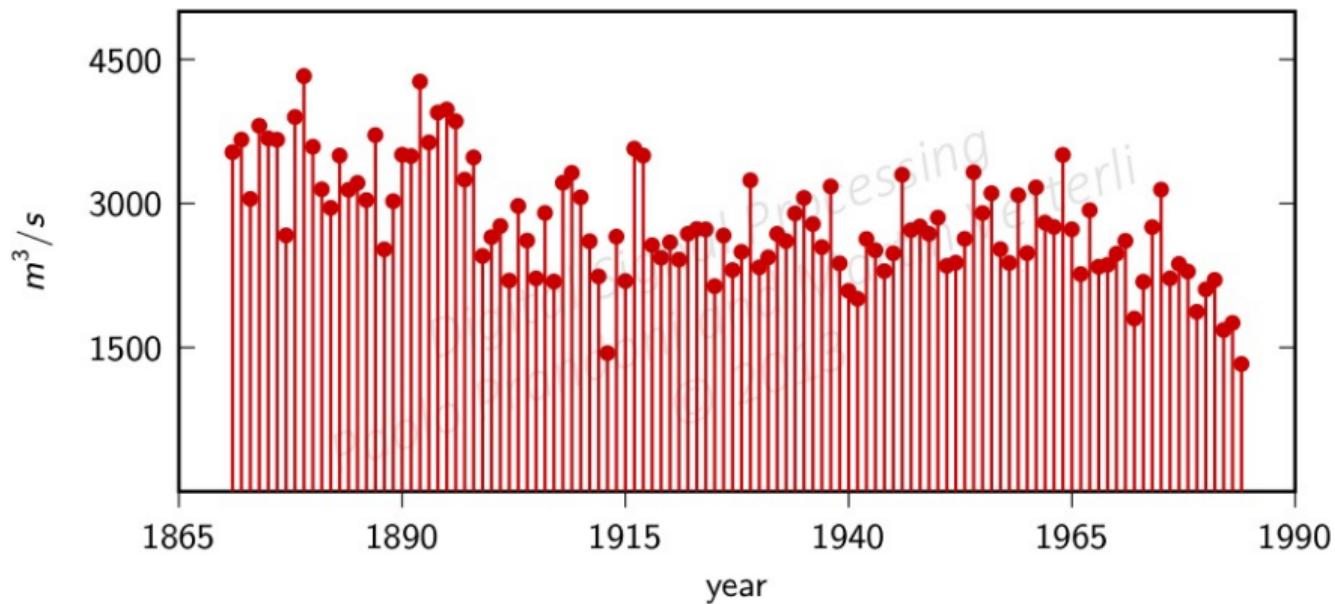
- ▶ discrete-time signals
- ▶ signal classes
- ▶ elementary operators
- ▶ shifts
- ▶ energy and power

Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

Meteorology (limnology): the floods of the Nile



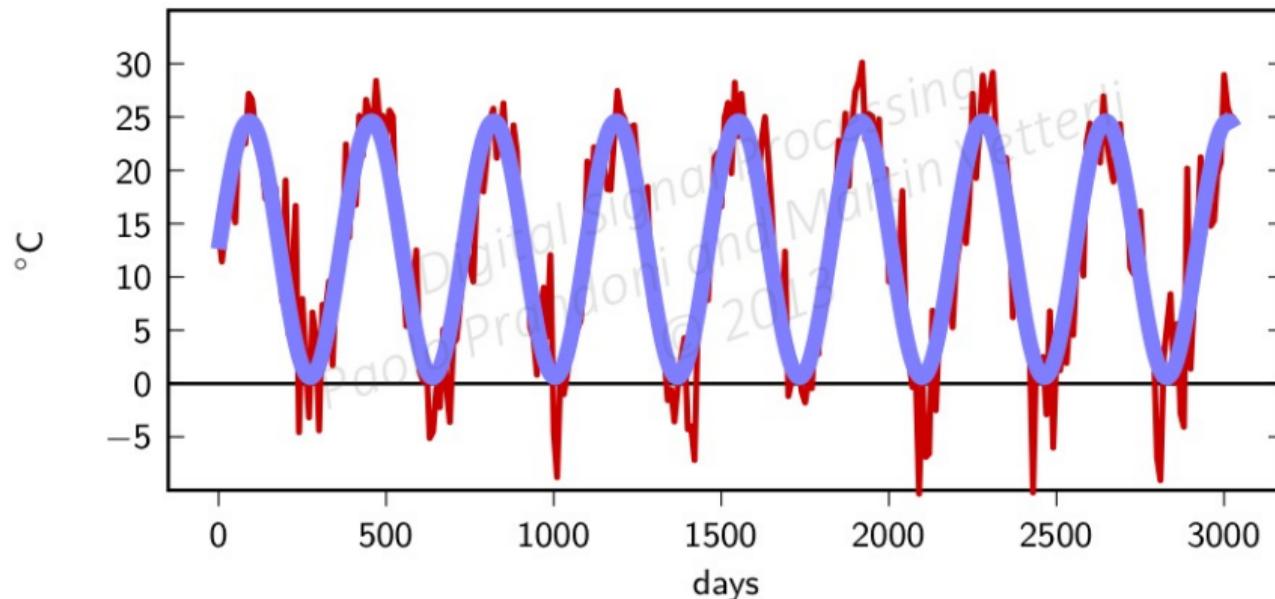
Representations of flood data: circa 2500 BC



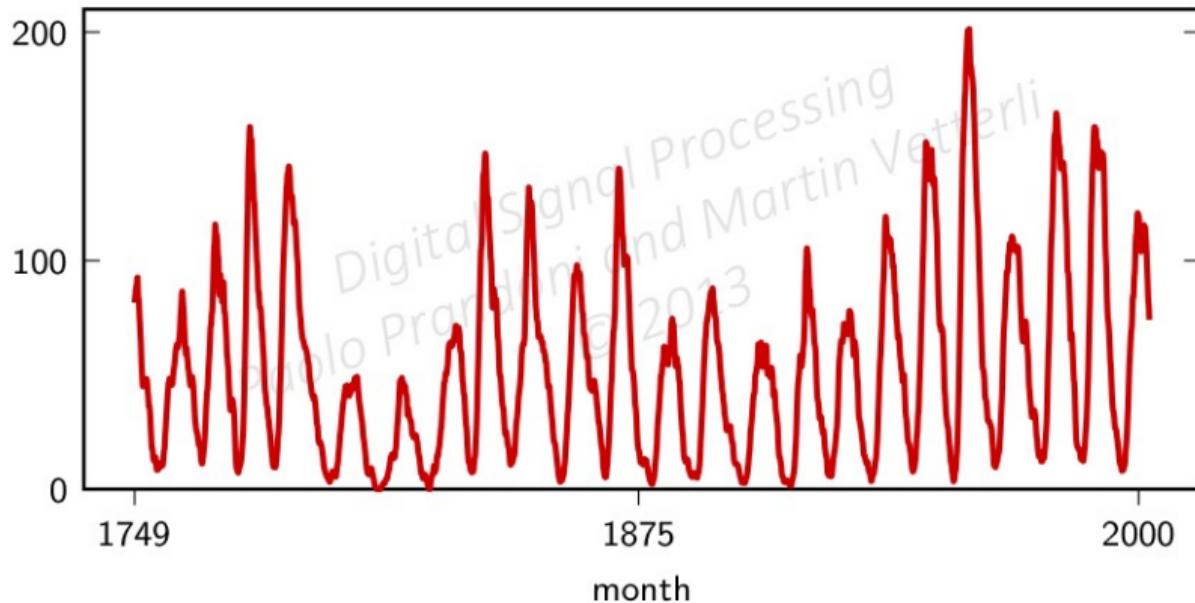
Representations of flood data: circa AD 2000

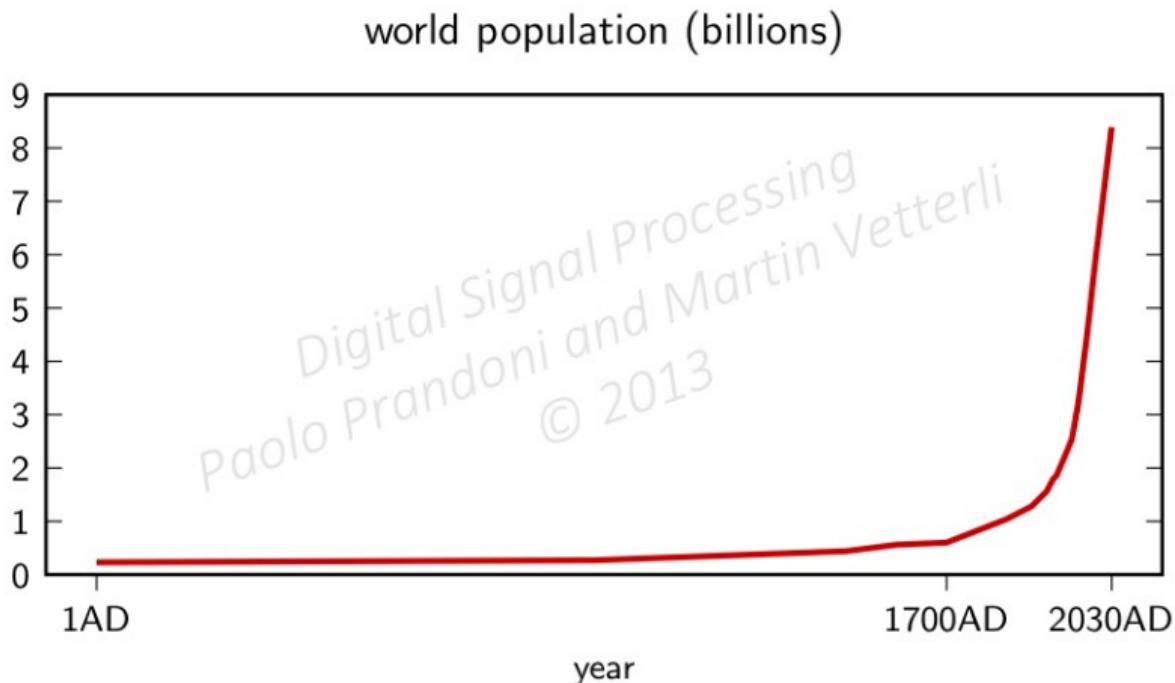
Probably your first scientific experiment...

Daily temperature

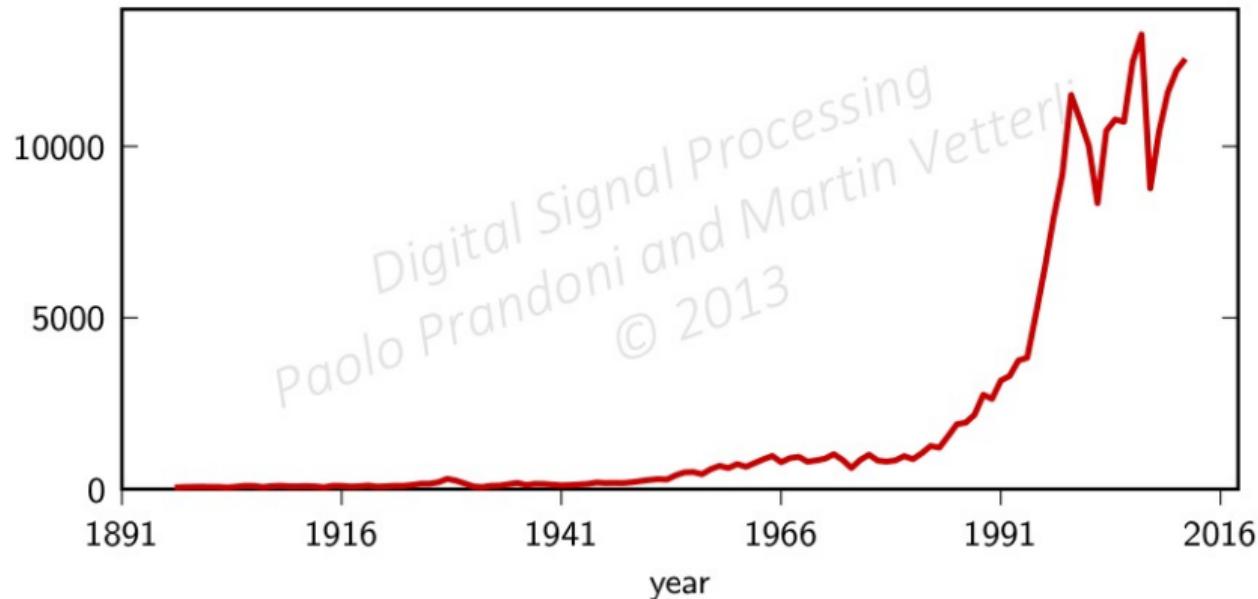


monthly solar spot activity, 1749 to 2003





a purely man-made signal: the Dow Jones industrial average



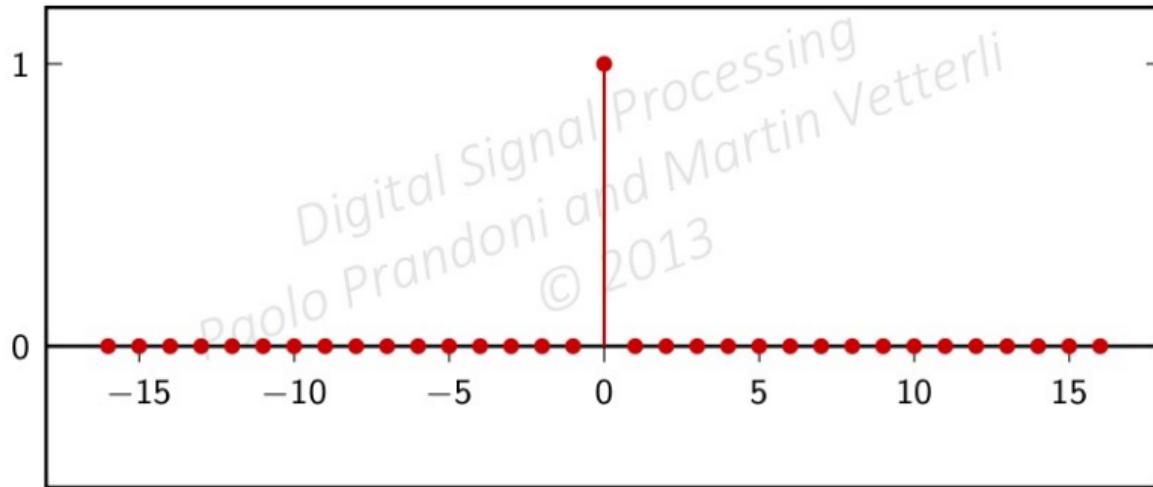
Discrete-time signal: a sequence of **complex** numbers

- ▶ one dimension (for now)
- ▶ notation:  $x[n]$
- ▶ two-sided sequences:  $x : \mathbb{Z} \rightarrow \mathbb{C}$
- ▶  $n$  is dimension-less “time”
- ▶ analysis: periodic measurement
- ▶ synthesis: stream of generated samples

Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

# The delta signal

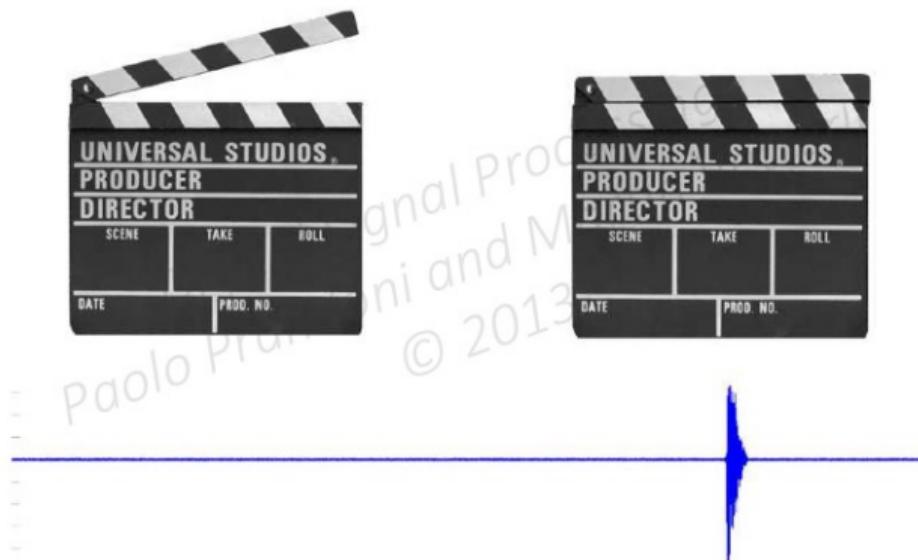
$$x[n] = \delta[n]$$



How do you synchronize audio and video...

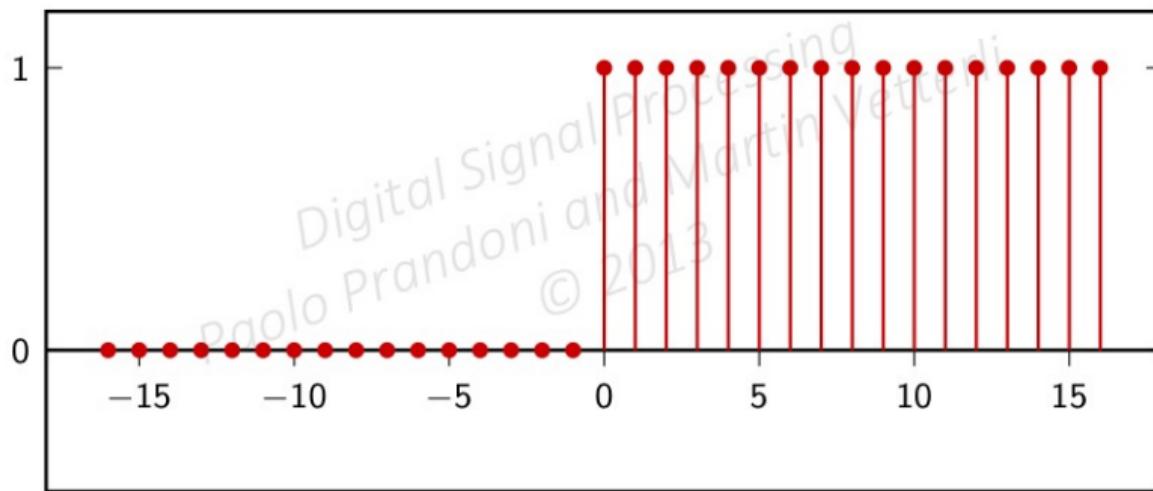


# How do you synchronize audio and video...



# The unit step

$$x[n] = u[n]$$

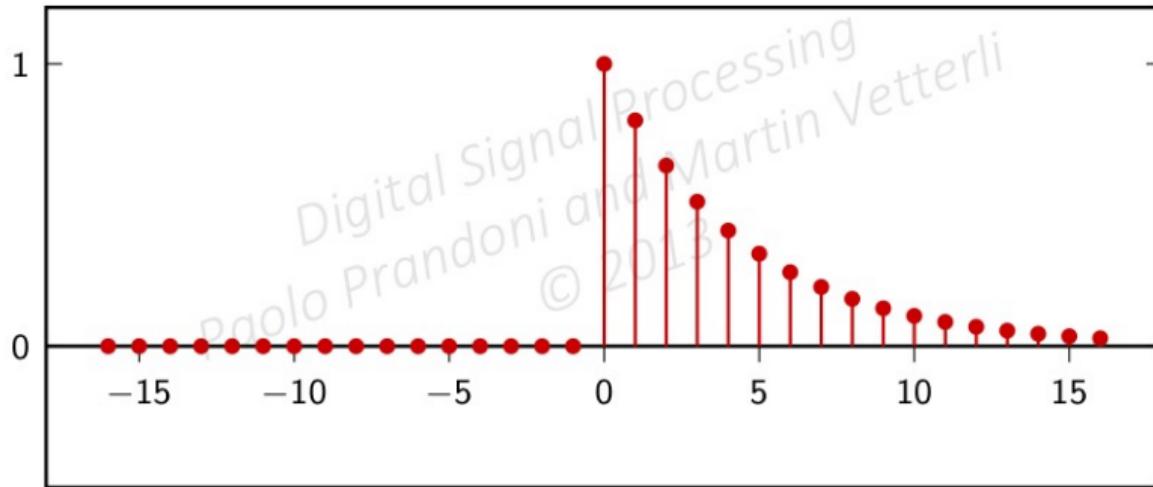


# The Frankenstein switch...



# The exponential decay

$$x[n] = |a|^n u[n], \quad |a| < 1$$



How fast does your coffee get cold...



Newton's law of cooling:

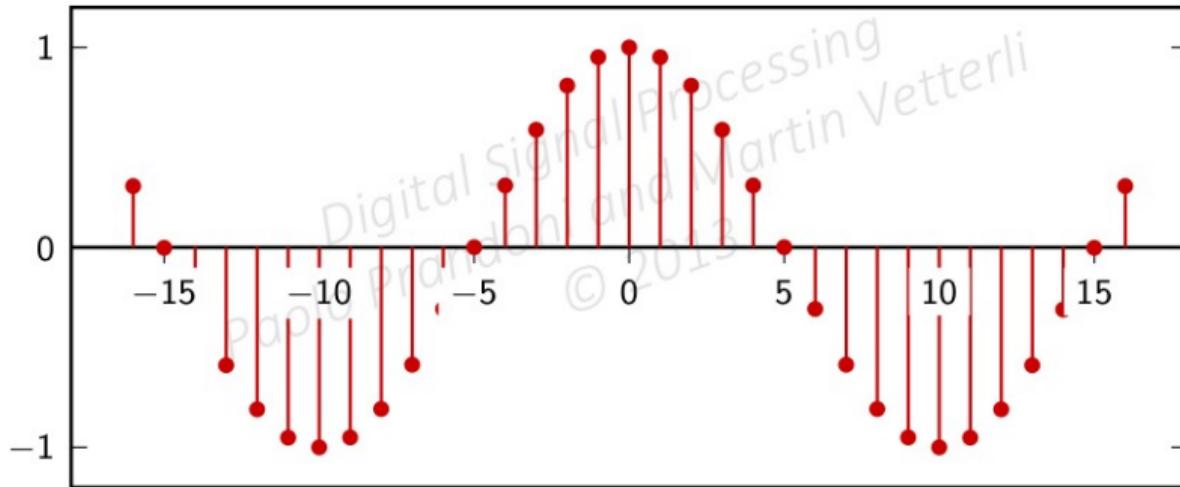
$$\frac{dT}{dt} = -c(T - T_{\text{env}})$$

$$T(t) = T_{\text{env}} + (T_0 - T_{\text{env}})e^{-ct}$$

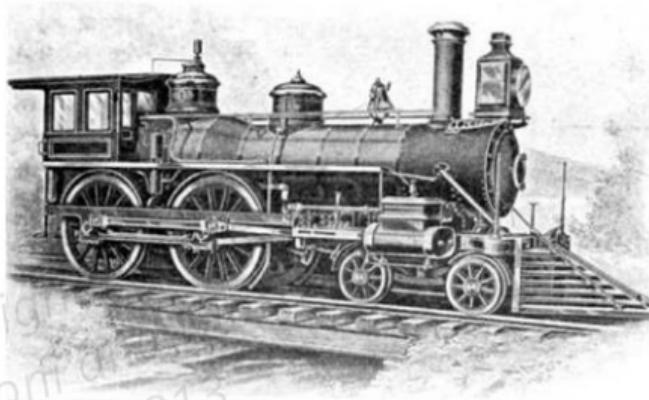
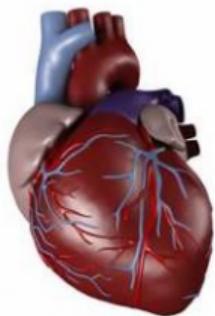
In practice:

- ▶ must have convection only
- ▶ must have large conductivity

$$x[n] = \sin(\omega_0 n + \theta)$$



# Oscillations are everywhere!



- ▶ finite-length
- ▶ infinite-length
- ▶ periodic
- ▶ finite-support

Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

- ▶ sequence notation:  $x[n], \quad n = 0, 1, \dots, N - 1$
- ▶ vector notation:  $\mathbf{x} = [x_0 \ x_1 \ \dots \ x_{N-1}]^T$
- ▶ practical entities, good for numerical packages (Matlab and the like)

- ▶ sequence notation:  $x[n], \quad n \in \mathbb{Z}$
- ▶ abstraction, good for theorems

Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

- ▶  $N$ -periodic sequence:  $\tilde{x}[n] = \tilde{x}[n + kN]$ ,  $n, k, N \in \mathbb{Z}$
- ▶ same information as finite-length of length  $N$
- ▶ “natural” bridge between finite and infinite lengths

- ▶ Finite-support sequence:

$$\bar{x}[n] = \begin{cases} x[n] & \text{if } 0 \leq n < N \\ 0 & \text{otherwise} \end{cases} \quad n \in \mathbb{Z}$$

- ▶ same information as finite-length of length  $N$
- ▶ another bridge between finite and infinite lengths

- ▶ scaling:

$$y[n] = \alpha x[n]$$

- ▶ sum:

$$y[n] = x[n] + z[n]$$

- ▶ product:

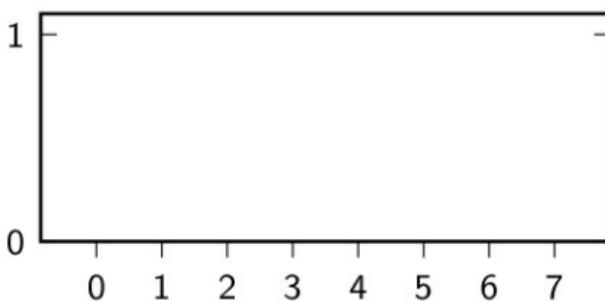
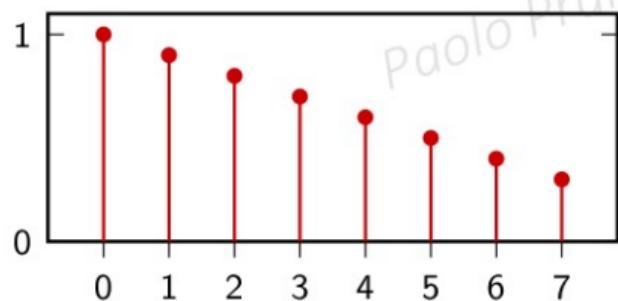
$$y[n] = x[n] \cdot z[n]$$

- ▶ shift by  $k$  (delay):

$$y[n] = x[n - k]$$

# Shift of a finite-length: finite-support

$$[x_0 \ x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7]$$

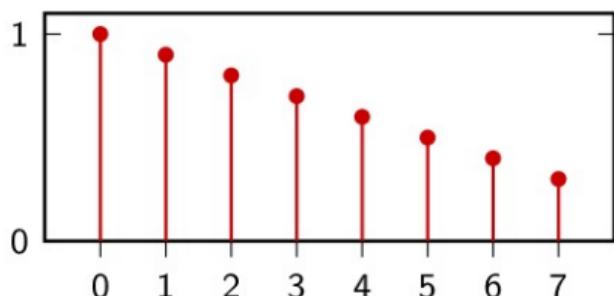
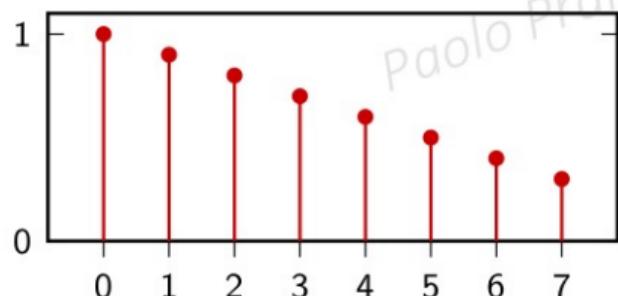


Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

# Shift of a finite-length: finite-support

 $x[n]$ 

...  $x_0 \ x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7 \ ...$

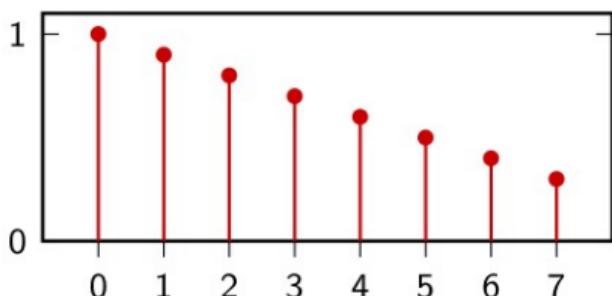
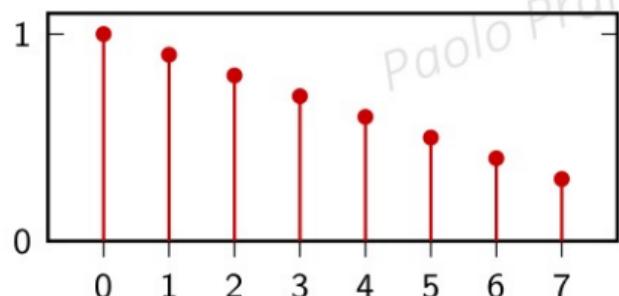


Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

# Shift of a finite-length: finite-support

 $\bar{x}[n]$ 

... 0 0 0  $x_0 \ x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7$  0 0 0 ...

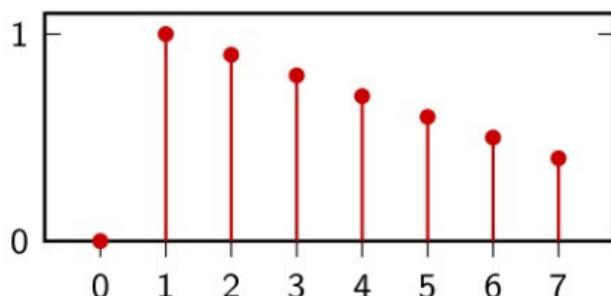


Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

# Shift of a finite-length: finite-support

$$\bar{x}[n - 1]$$

$$\dots \ 0 \ 0 \ 0 \ 0 \ x_0 \ x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7 \ 0 \ 0 \ \dots$$

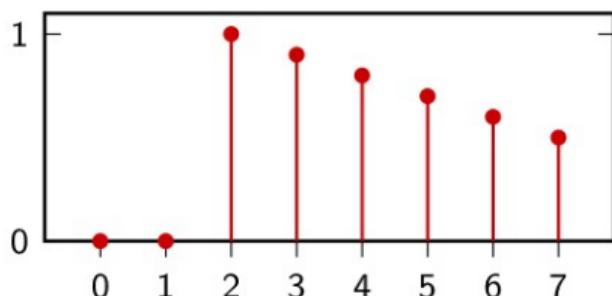
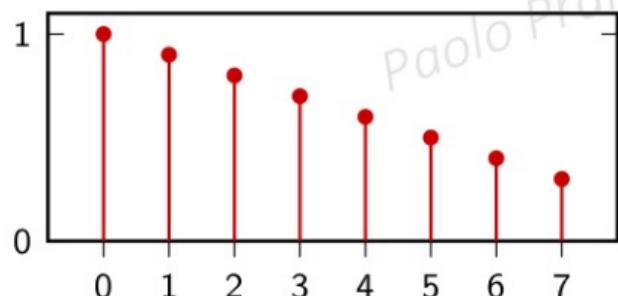


Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

# Shift of a finite-length: finite-support

$$\bar{x}[n - 2]$$

... 0 0 0 0 0 0  $x_0$   $x_1$   $x_2$   $x_3$   $x_4$   $x_5$   $x_6$   $x_7$  0 ...

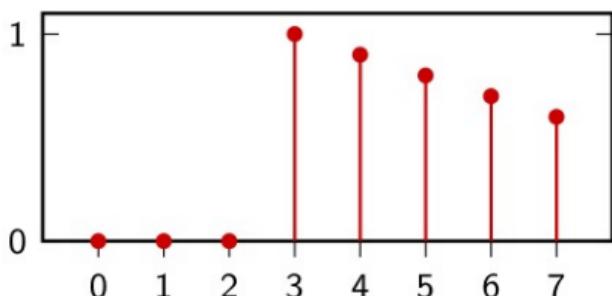
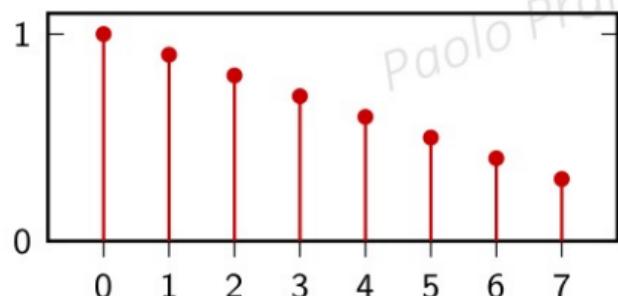


Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

# Shift of a finite-length: finite-support

$$\bar{x}[n - 3]$$

$$\dots \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ x_0 \ x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7 \ \dots$$

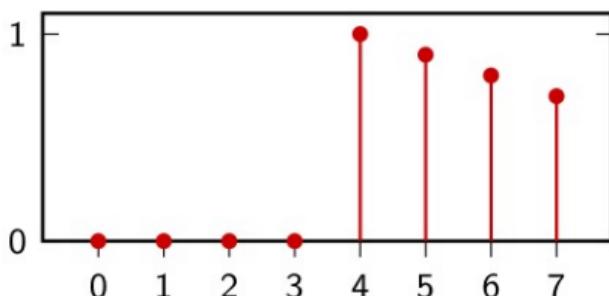
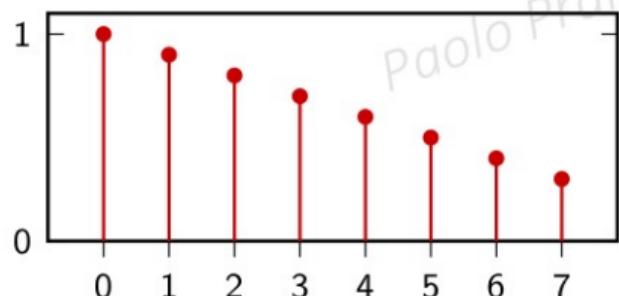


Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

# Shift of a finite-length: finite-support

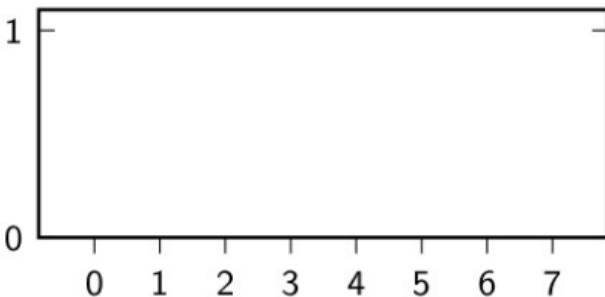
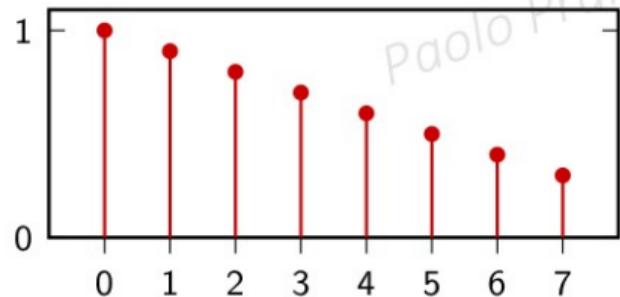
$$\bar{x}[n - 4]$$

... 0 0 0 0 0 0  $x_0$   $x_1$   $x_2$   $x_3$   $x_4$   $x_5$   $x_6$  ...



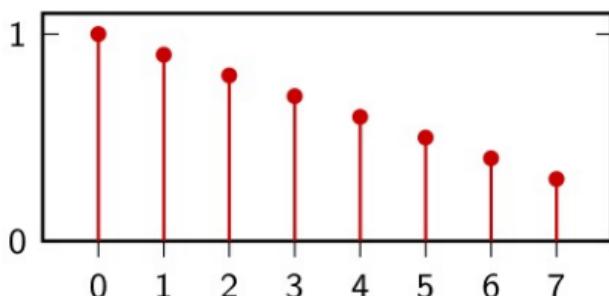
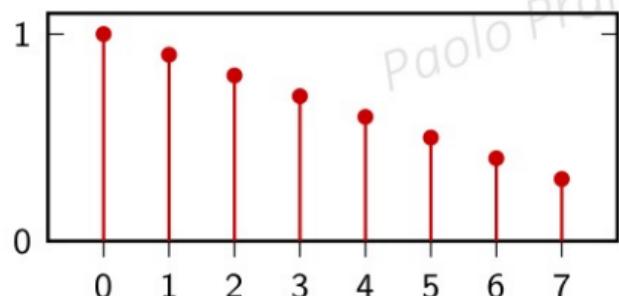
# Shift of a finite-length: periodic extension

$$[x_0 \ x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7]$$



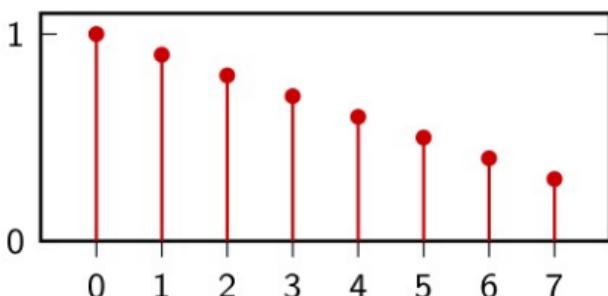
Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

# Shift of a finite-length: periodic extension

 $x[n]$  $\dots \boxed{x_0 \ x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7} \dots$ 

Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

# Shift of a finite-length: periodic extension

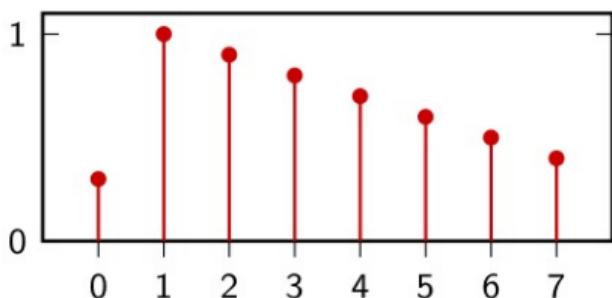
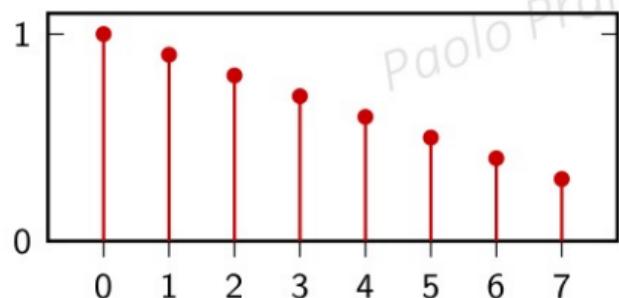
 $\tilde{x}[n]$  $\dots \quad x_5 \quad x_6 \quad x_7 \quad x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_0 \quad x_1 \quad x_2 \quad \dots$ 

Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

# Shift of a finite-length: periodic extension

$$\tilde{x}[n - 1]$$

$\dots \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_0 \quad x_1 \quad \dots$

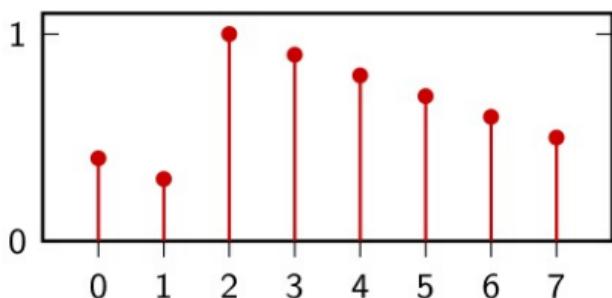
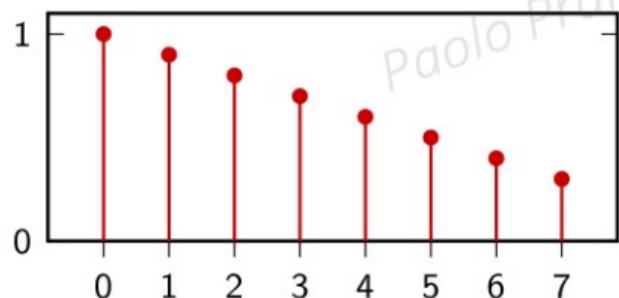


Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

# Shift of a finite-length: periodic extension

$$\tilde{x}[n - 2]$$

$\dots \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_0 \quad \dots$

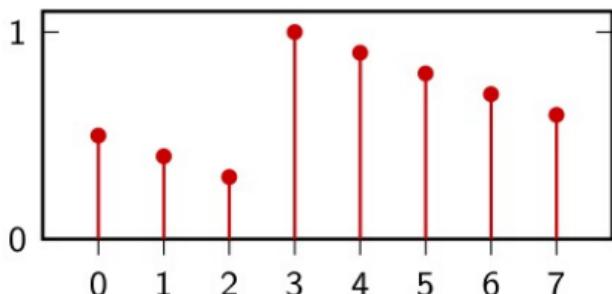
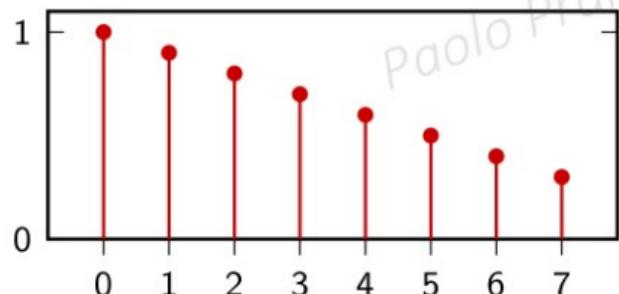


Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

# Shift of a finite-length: periodic extension

$$\tilde{x}[n - 3]$$

$\dots \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad \dots$

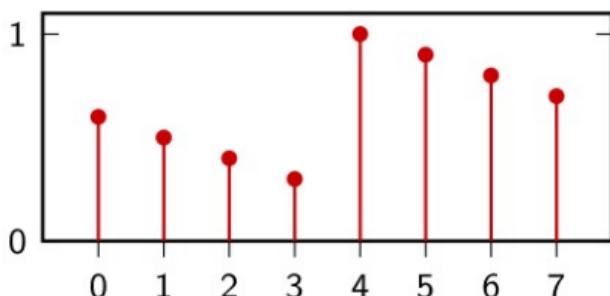
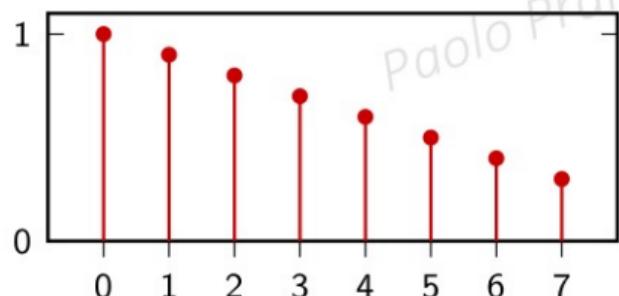


Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

# Shift of a finite-length: periodic extension

$$\tilde{x}[n - 4]$$

$\dots \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad \dots$



Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

$$E_x = \sum_{n=-\infty}^{\infty} |x[n]|^2$$

$$P_x = \lim_{N \rightarrow \infty} \frac{1}{2N+1} \sum_{n=-N}^N |x[n]|^2$$

$$E_{\tilde{x}} = \infty$$

$$P_{\tilde{x}} \equiv \frac{1}{N} \sum_{n=0}^{N-1} |\tilde{x}[n]|^2$$

Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

# Digital Signal Processing

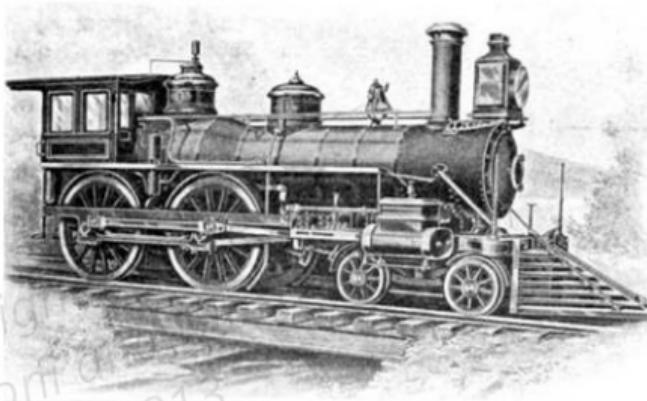
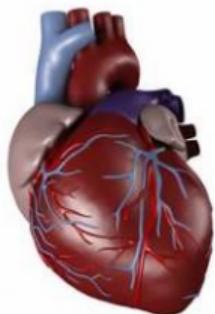
Module 2.2: the complex exponential

Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

- ▶ the complex exponential
- ▶ periodicity
- ▶ wagonwheel effect and maximum “speed”
- ▶ digital and real-world frequency

Digital Signal Processing  
P. P. Vaidyanathan and Martin Vetterli  
© 2013

# Oscillations are everywhere



Ingredients:

- ▶ a frequency  $\omega$  (units: radians)
- ▶ an initial phase  $\phi$  (units: radians)
- ▶ an amplitude  $A$  (units depending on underlying measurement)
- ▶ a trigonometric function

e.g.  $x[n] = A \cos(\omega n + \phi)$

the trigonometric function of choice in DSP is the complex exponential:

$$\begin{aligned}x[n] &= Ae^{j(\omega n + \phi)} \\&= A[\cos(\omega n + \phi) + j \sin(\omega n + \phi)]\end{aligned}$$

- ▶ makes sense: sines and cosines always go together
- ▶ simpler math: trigonometry becomes algebra
- ▶ we can use complex numbers in digital systems

Example: change the phase of a pure cosine

$$\cos(\omega n + \phi) = a \cos(\omega n) + b \sin(\omega n), \quad a = \cos \phi, \quad b = \sin \phi$$

- ▶ each sinusoid is always a sum of sine and cosine
- ▶ we have to remember complex trigonometric formulas
- ▶ we have to carry more terms in our equations

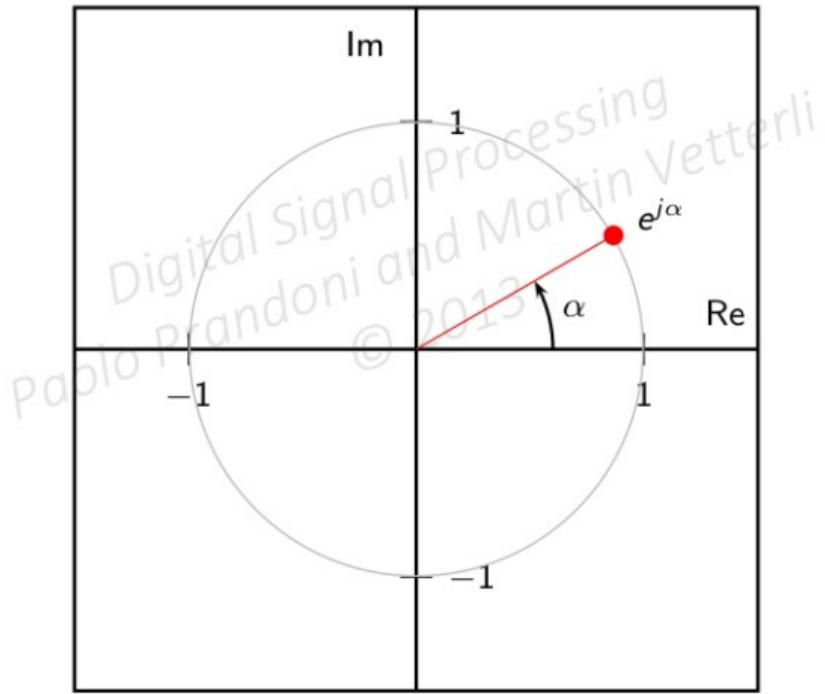
Example: change the phase of a pure cosine

$$\operatorname{Re}\{e^{j(\omega n + \phi)}\} = \operatorname{Re}\{e^{j\omega n} e^{j\phi}\}$$

- ▶ sine and cosine “live” together
- ▶ phase shift is simple multiplication
- ▶ notation is simpler

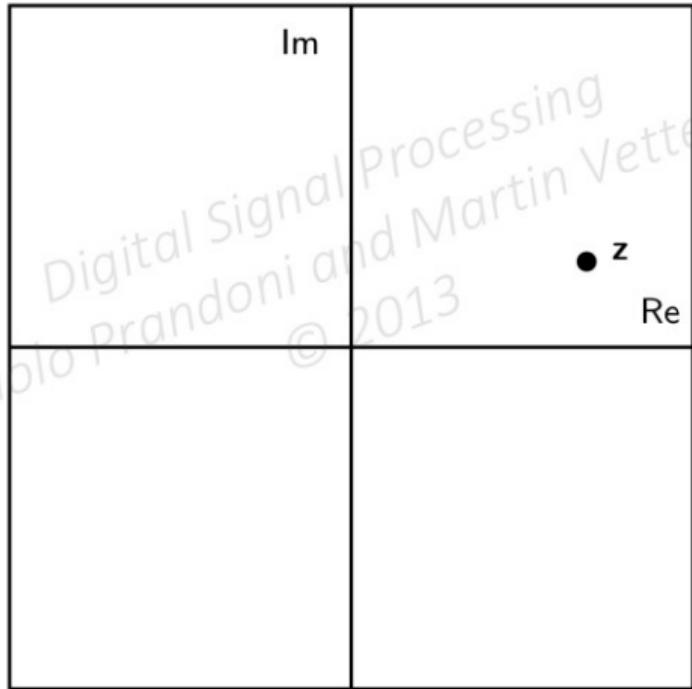
# The complex exponential

$$e^{j\alpha} = \cos \alpha + j \sin \alpha$$



# The complex exponential

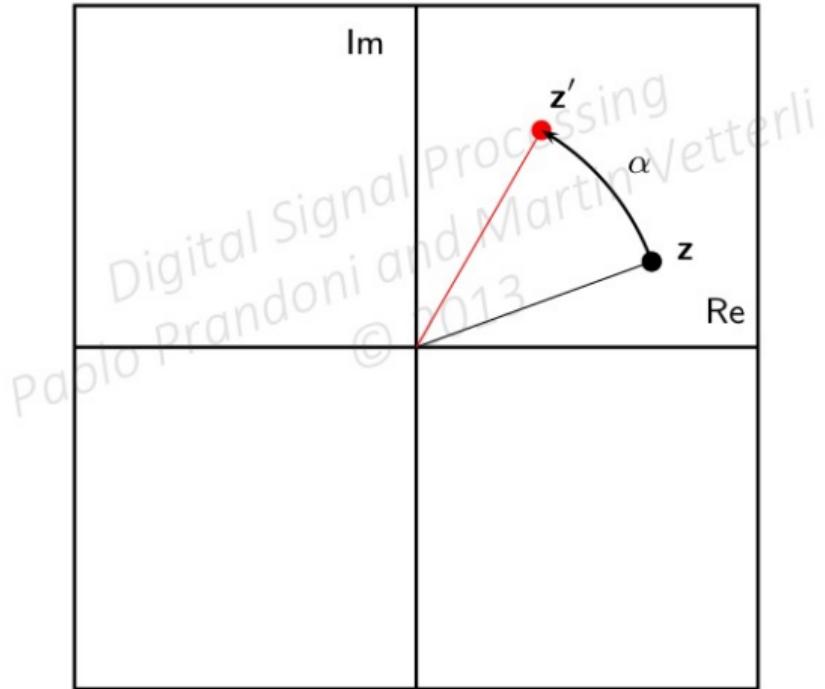
$z$ : point on the complex plane



Digital Signal Processing  
Pablo Prandoni and Martin Vetterli  
© 2013

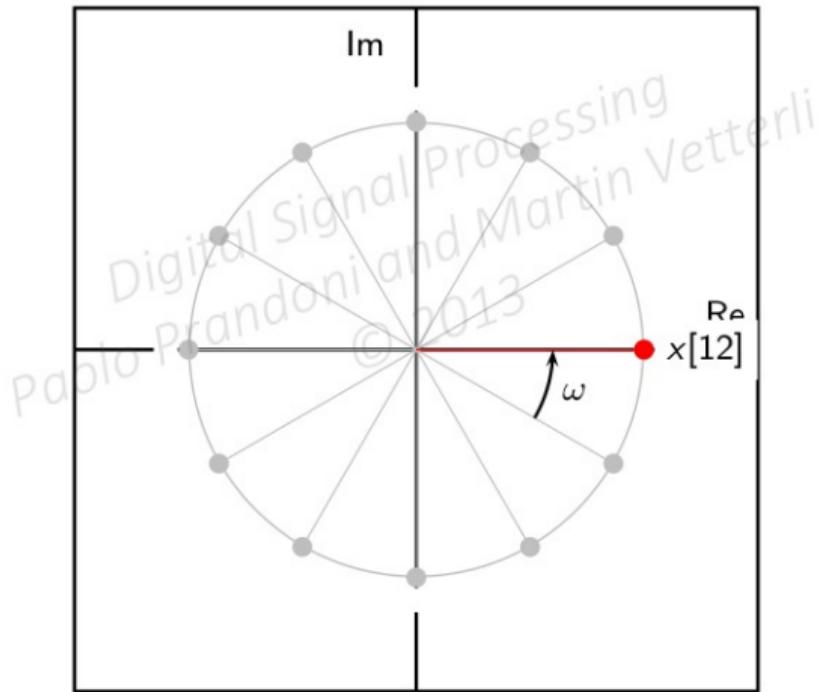
# The complex exponential

$$\text{rotation: } z' = z e^{j\alpha}$$



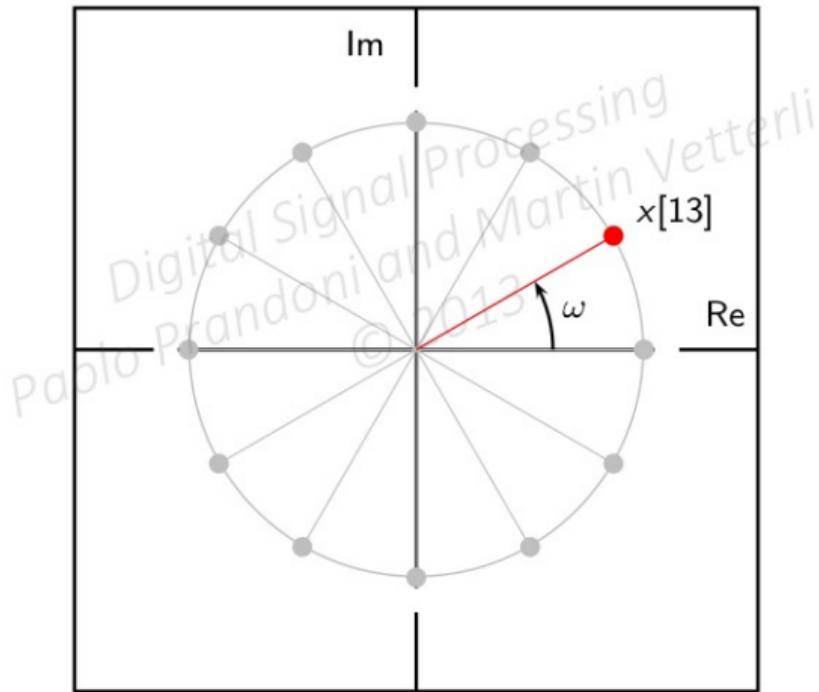
# The complex exponential generating machine

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$

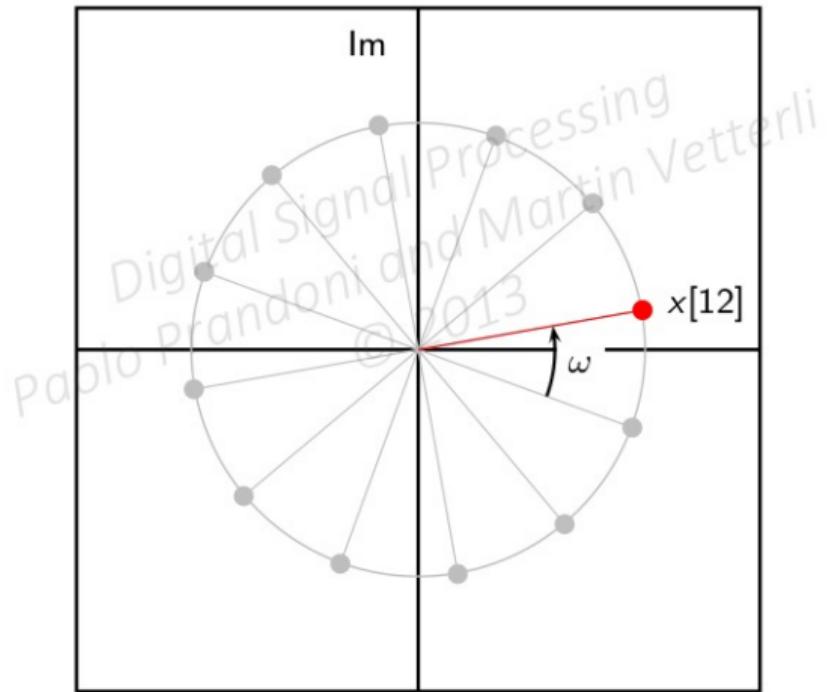


# The complex exponential generating machine

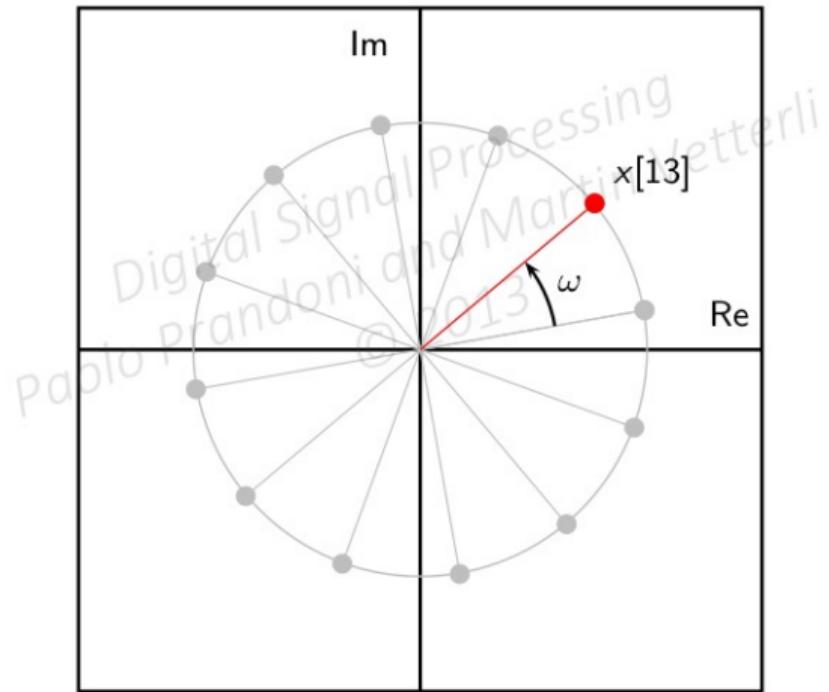
$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



$$x[n] = e^{j(\omega n + \phi)}; \quad x[n+1] = e^{j\omega} x[n], \quad x[0] = e^{j\phi}$$

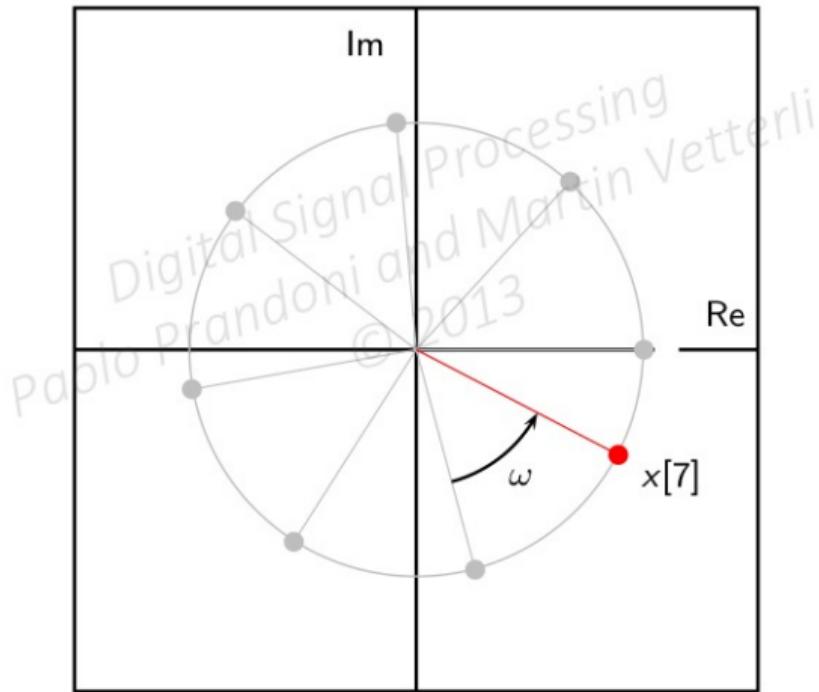


$$x[n] = e^{j(\omega n + \phi)}; \quad x[n+1] = e^{j\omega} x[n], \quad x[0] = e^{j\phi}$$



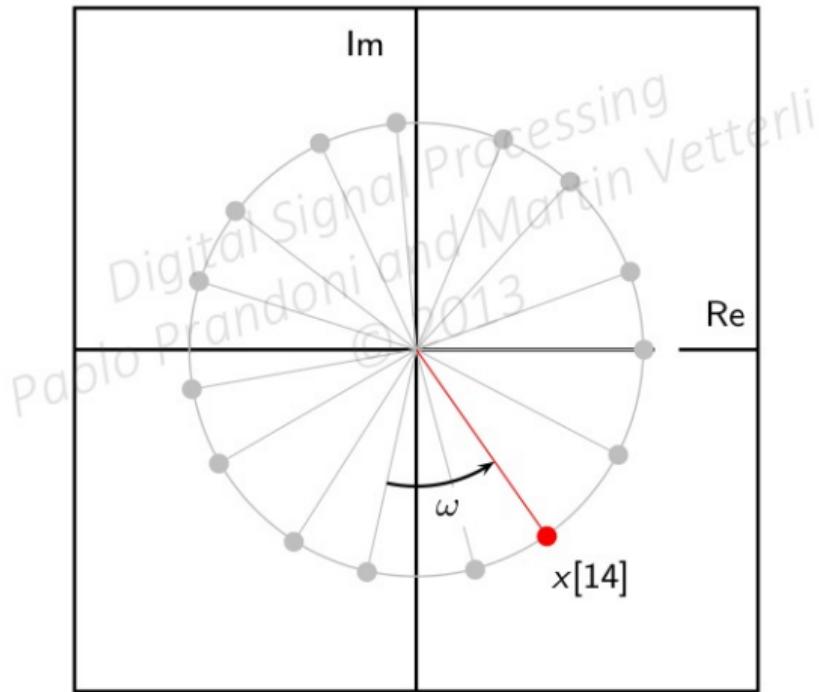
# Careful: not every sinusoid is periodic in discrete time

$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



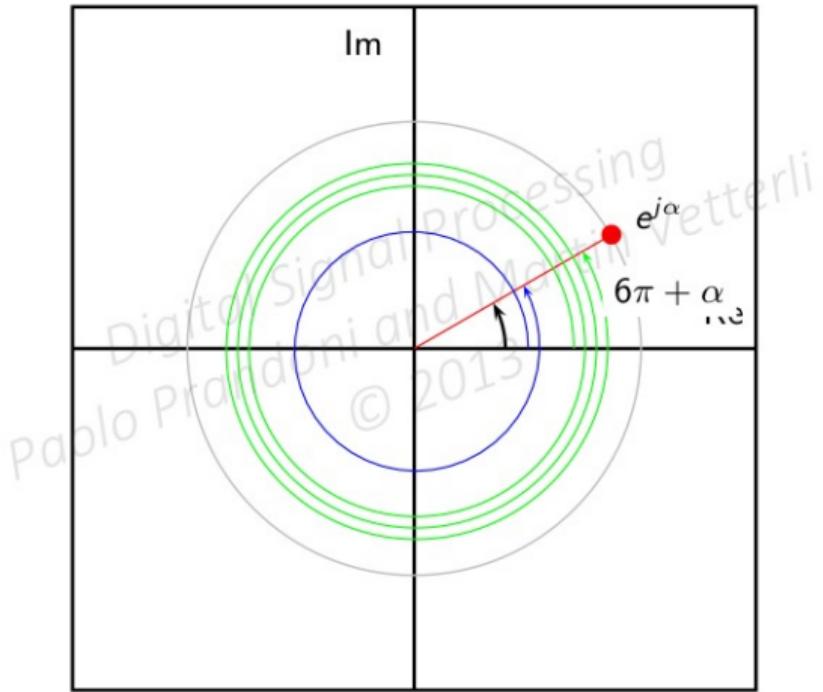
# Careful: not every sinusoid is periodic in discrete time

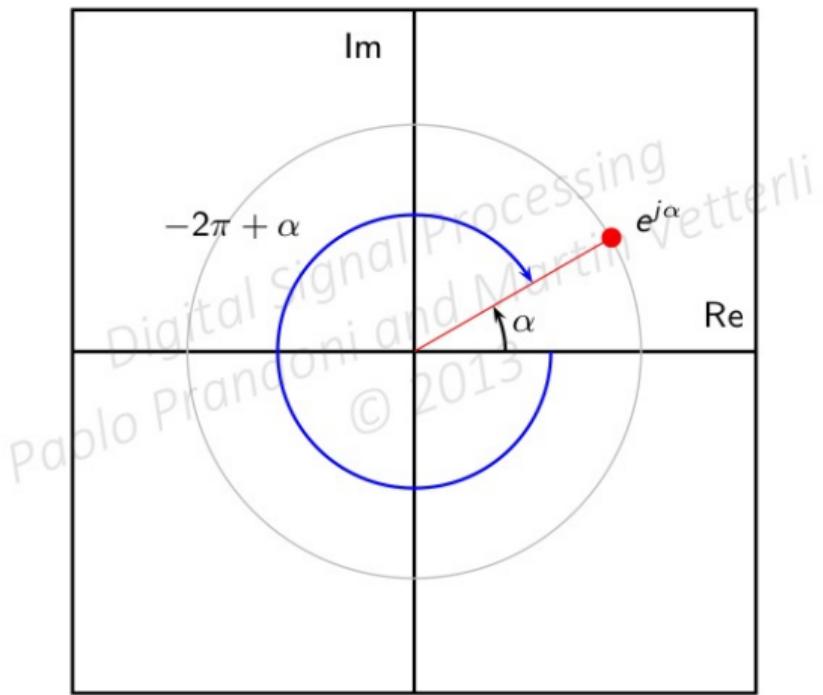
$$x[n] = e^{j\omega n}; \quad x[n+1] = e^{j\omega} x[n]$$



$$e^{j\omega n} \text{ periodic} \iff \omega = \frac{M}{N} 2\pi, M, N \in \mathbb{N}$$

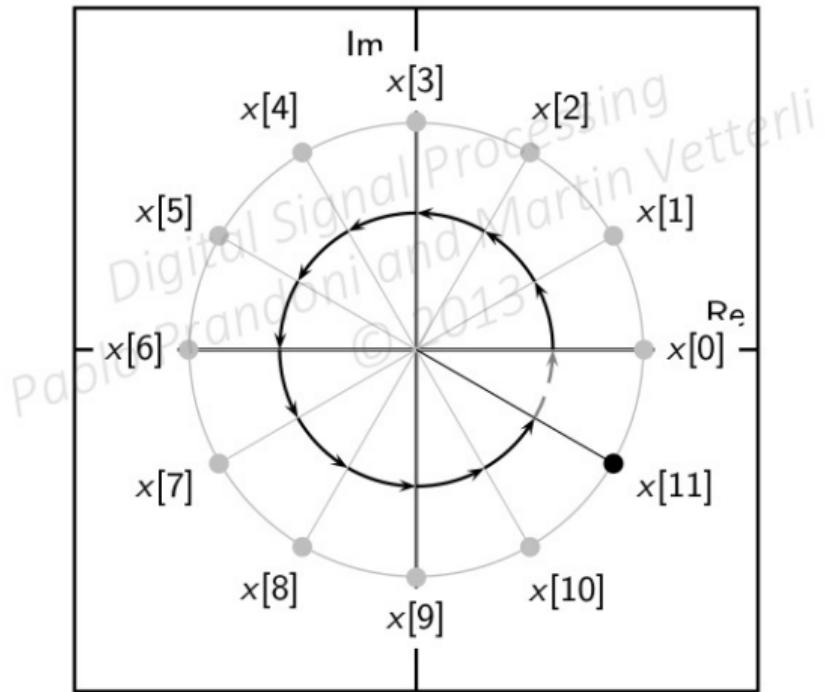
$$e^{j\omega} = e^{j(\omega+2k\pi)} \quad \forall k \in \mathbb{N}$$





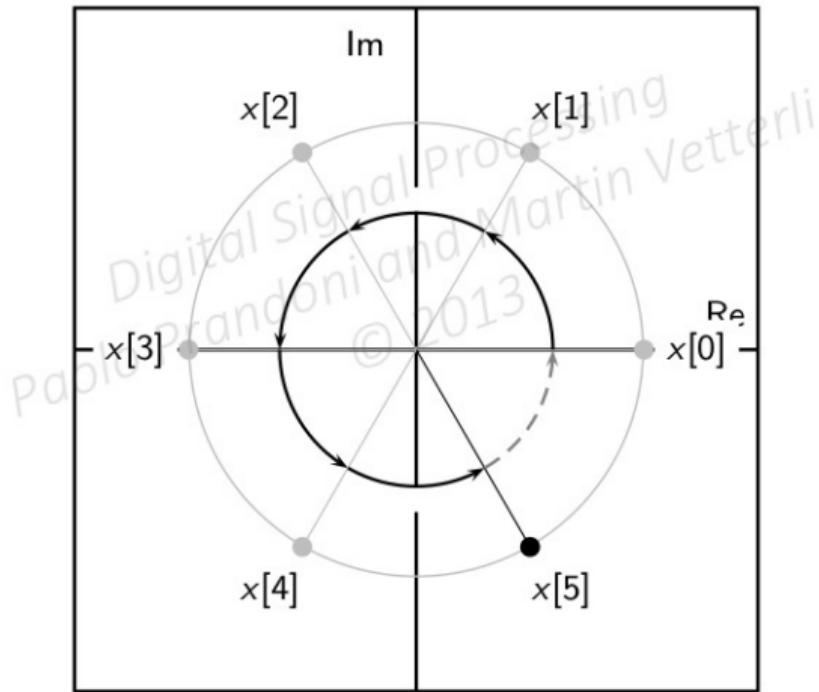
# How “fast” can we go?

$$\omega = 2\pi/12$$



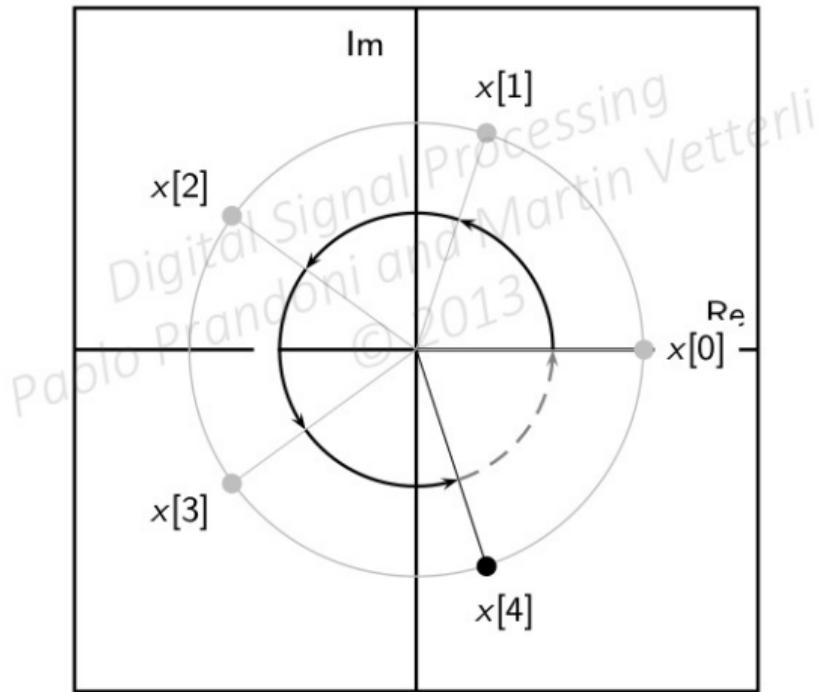
# How “fast” can we go?

$$\omega = 2\pi/6$$



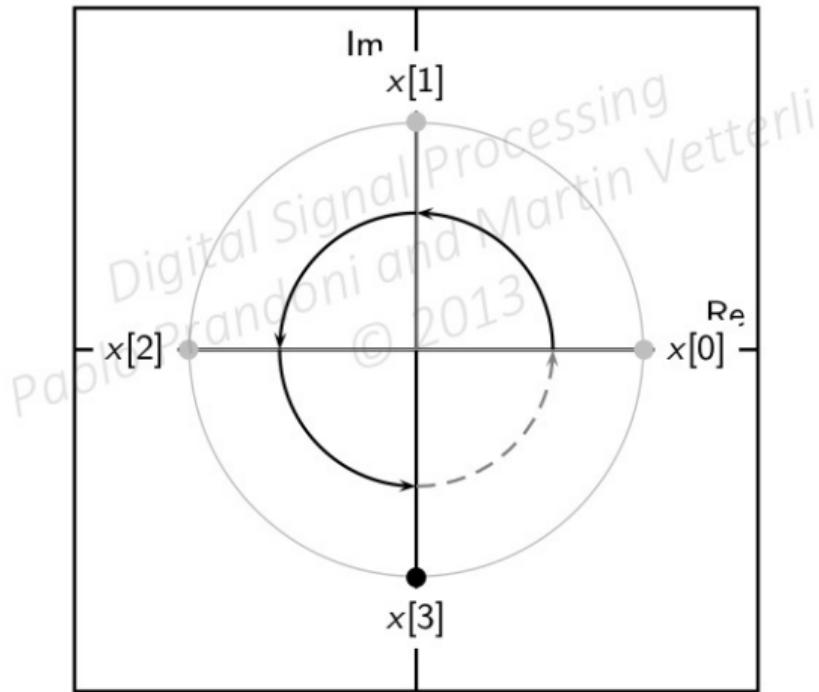
# How “fast” can we go?

$$\omega = 2\pi/5$$



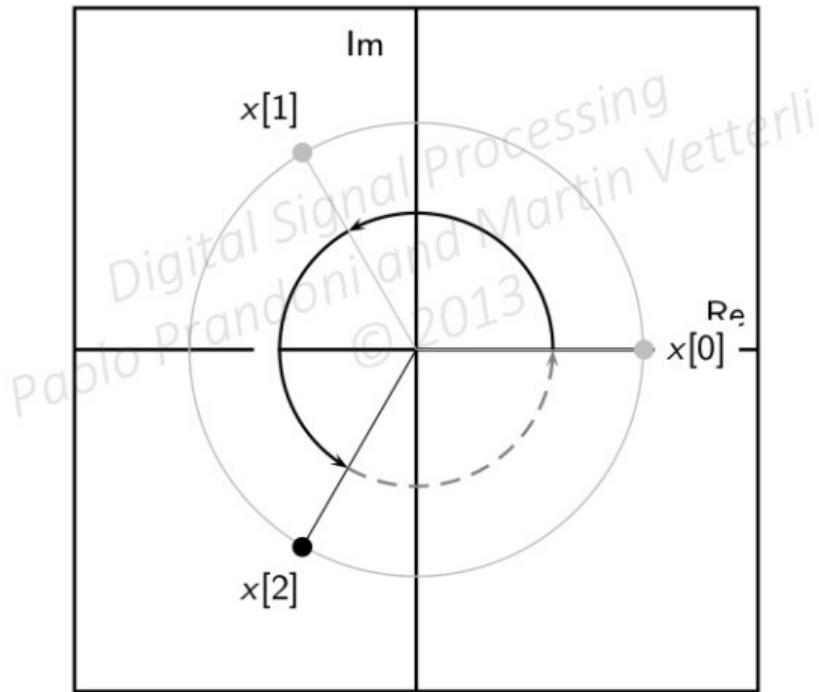
# How “fast” can we go?

$$\omega = 2\pi/4$$



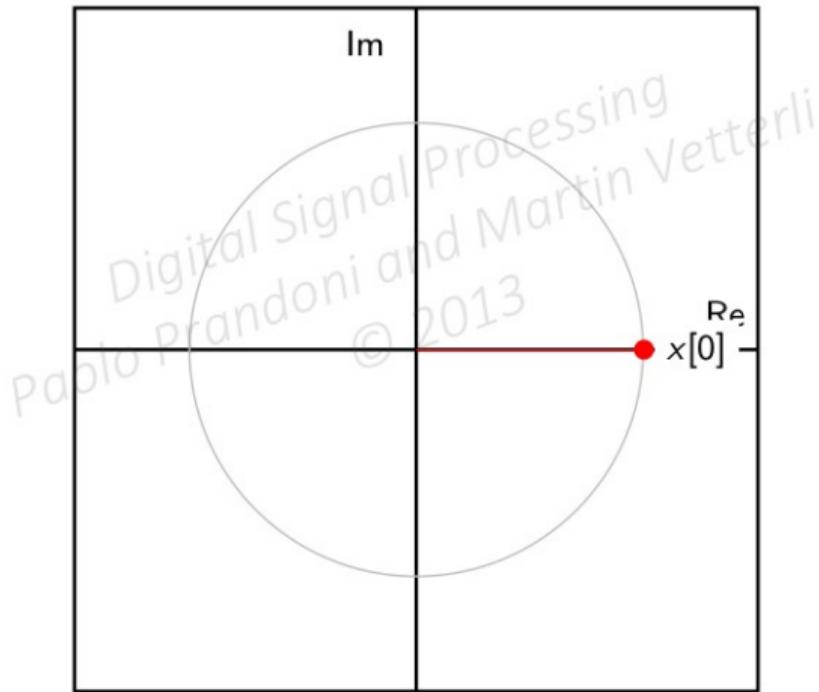
# How “fast” can we go?

$$\omega = 2\pi/3$$



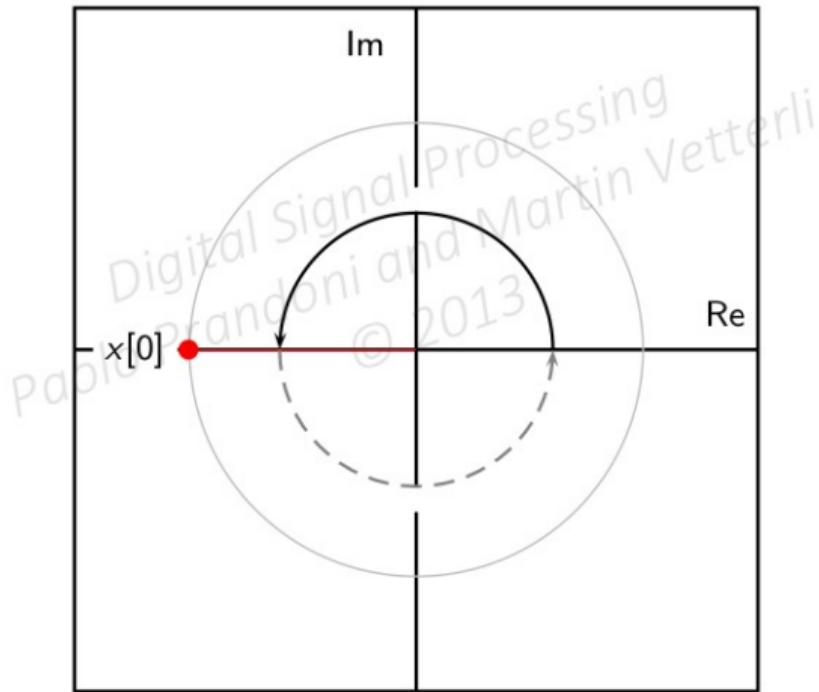
# How “fast” can we go?

$$\omega = 2\pi/2 = \pi$$



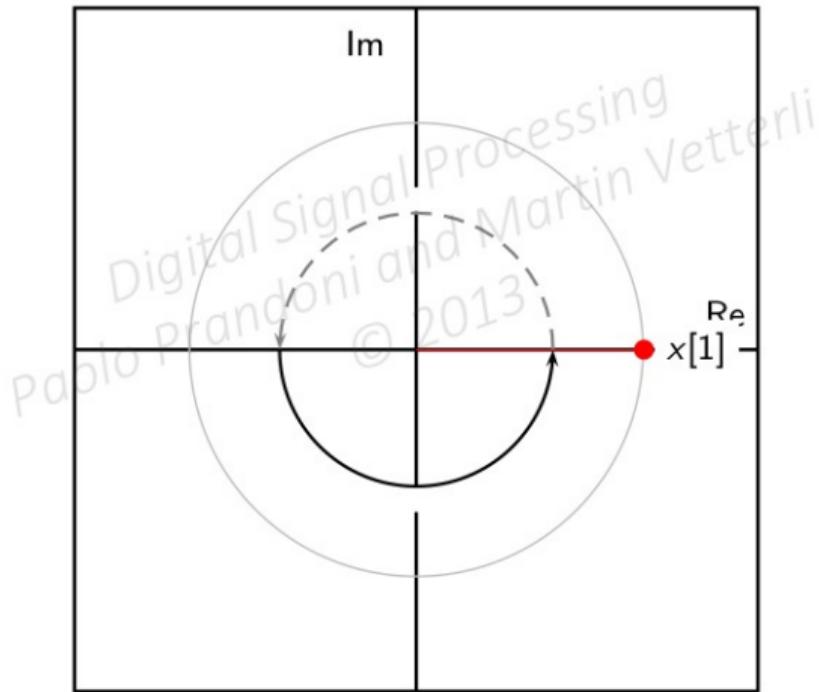
# How “fast” can we go?

$$\omega = 2\pi/2 = \pi$$



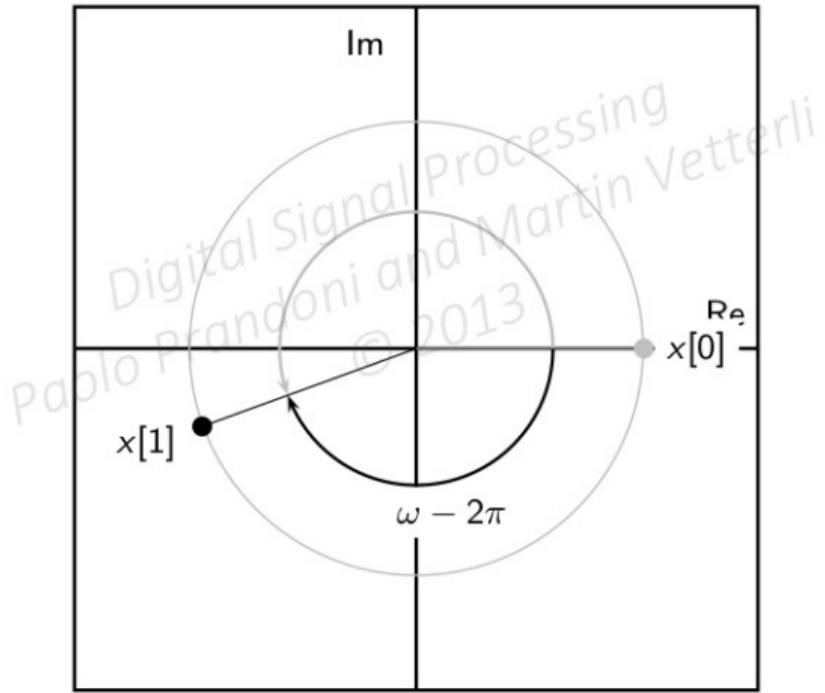
# How “fast” can we go?

$$\omega = 2\pi/2 = \pi$$

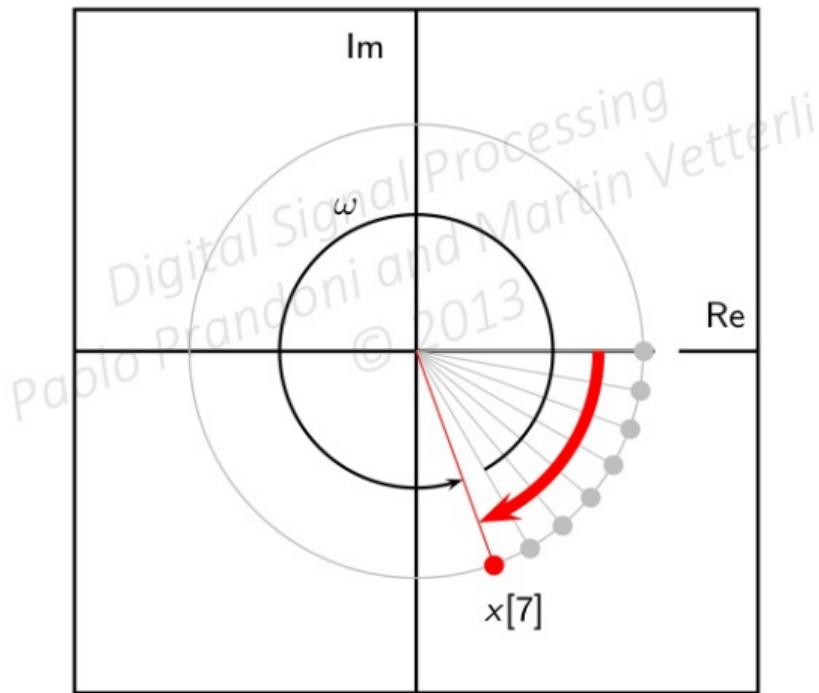


# What if we go “faster” ?

$$\pi < \omega < 2\pi$$



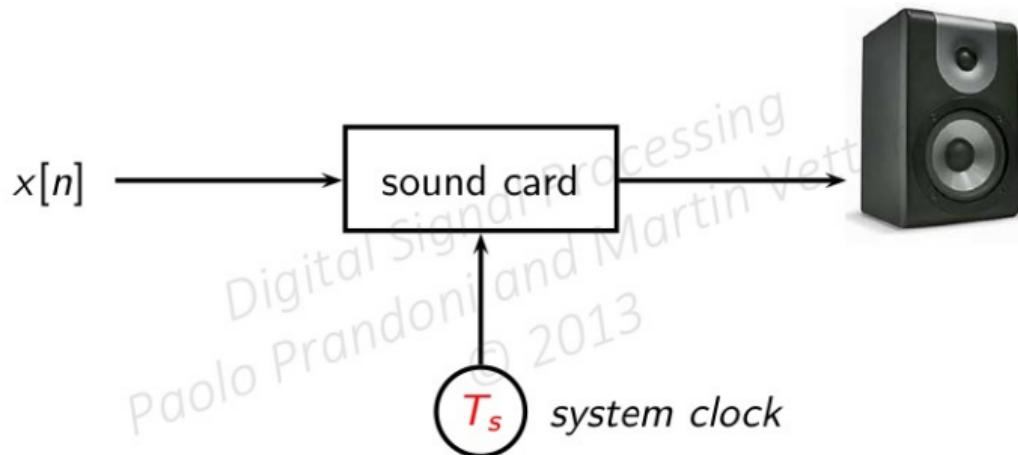
$$\omega = 2\pi - \alpha, \quad \alpha \text{ small}$$



Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

- ▶ Discrete time:
  - $n$ : no physical dimension (just a counter)
  - periodicity: how many samples before pattern repeats
  
- ▶ “Real world”:
  - periodicity: how many *seconds* before pattern repeats
  - frequency measured in Hz ( $s^{-1}$ )

# How your PC plays sounds



- ▶ set  $T_s$ , time in seconds between samples
- ▶ periodicity of  $M$  samples → periodicity of  $MT_s$  seconds
- ▶ real world frequency:

$$f = \frac{1}{MT_s}$$

Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
@ 2013

# Digital Signal Processing

Module 2.3: the Karplus-Strong algorithm

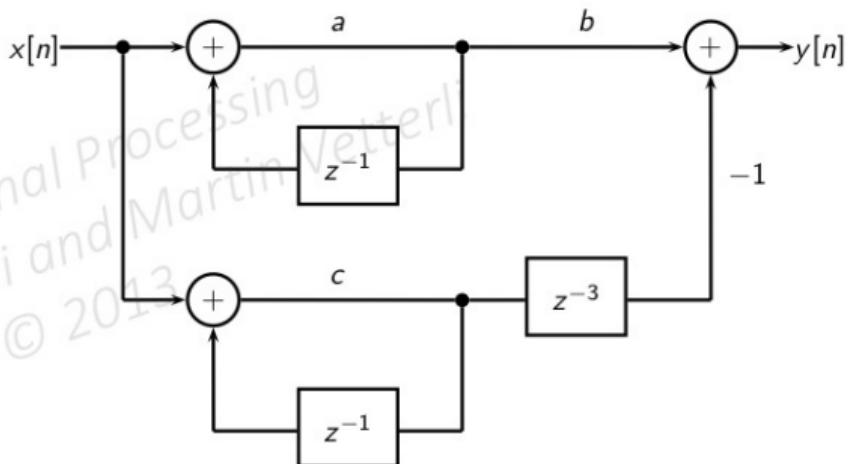
Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

- ▶ DSP building blocks
- ▶ moving averages and simple feedback loops
- ▶ a sound synthesizer

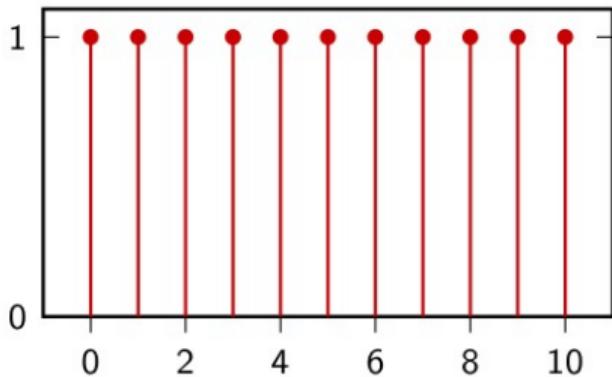
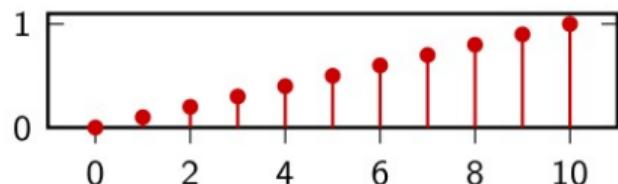
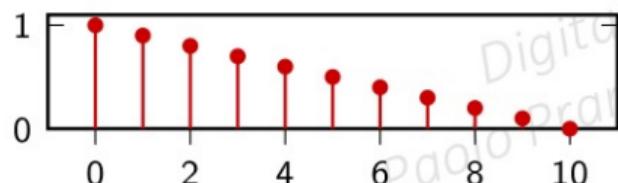
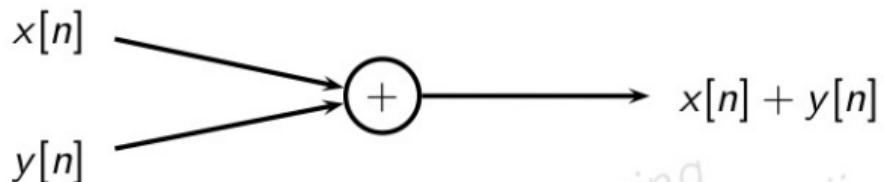
Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

- ▶ DSP as Lego: The fundamental building blocks
- ▶ Averages and moving averages
- ▶ Recursion: Revisiting your bank account
- ▶ Building a simple recursive synthesizer
- ▶ Examples of sounds

Digital Signal Processing  
Paolo Frasconi and Martin Vetterli  
© 2013

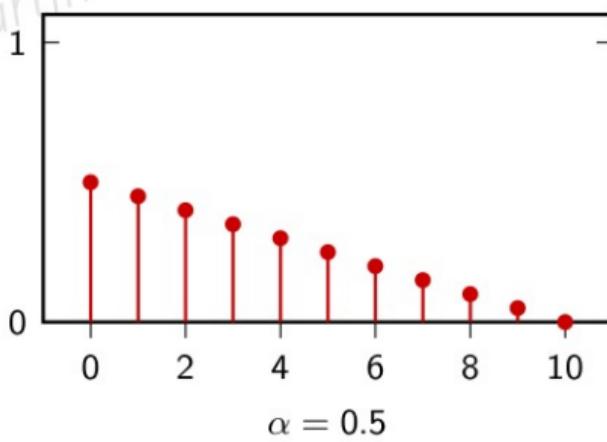
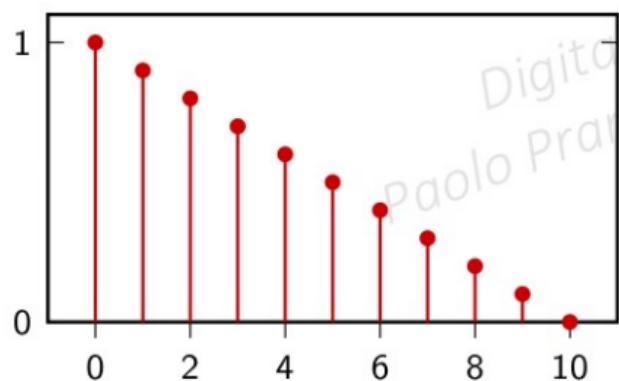


# Building Blocks: Adder



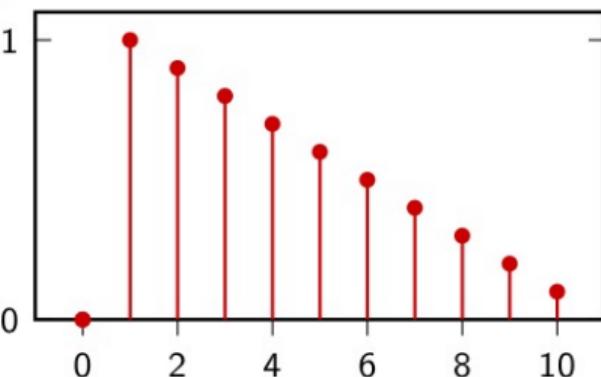
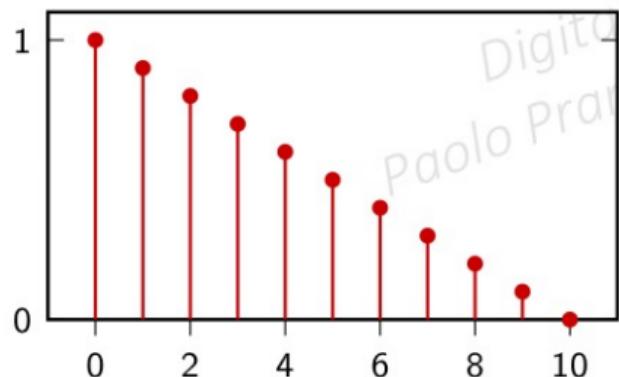
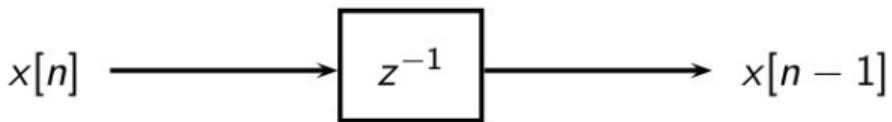
# Building Blocks: Multiplier

$$x[n] \xrightarrow{\alpha} \alpha x[n]$$

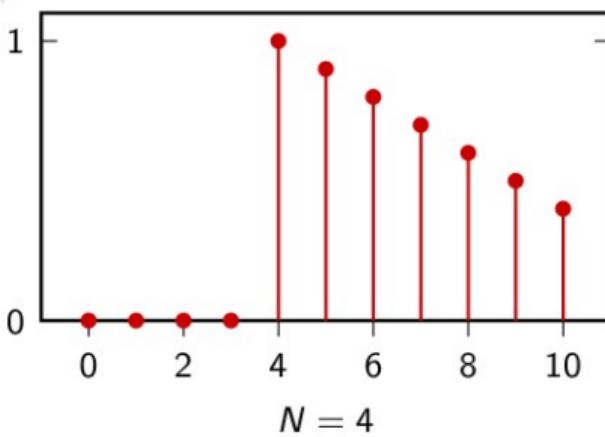
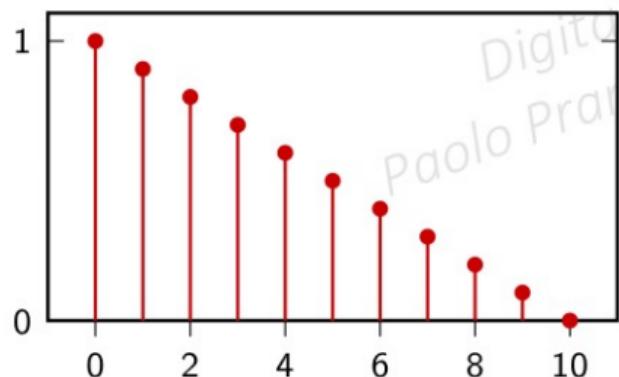
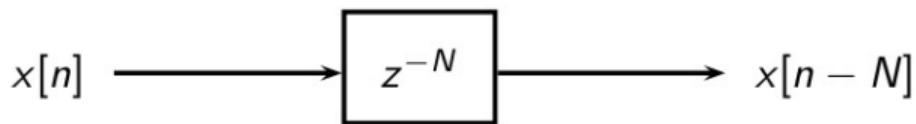


Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

# Building Blocks: Unit Delay



# Building Blocks: Arbitrary Delay



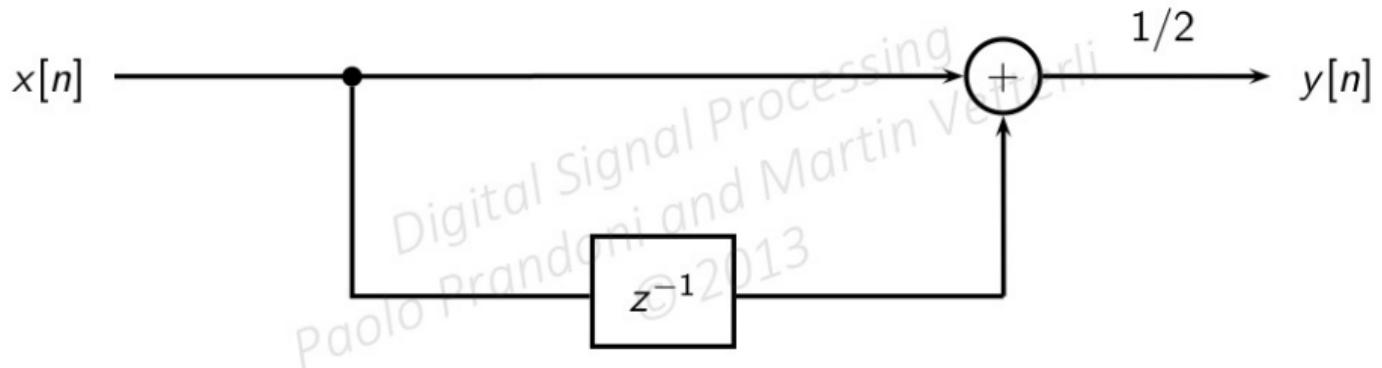
- ▶ simple average:

$$m = \frac{a + b}{2}$$

- ▶ moving average: take a “local” average

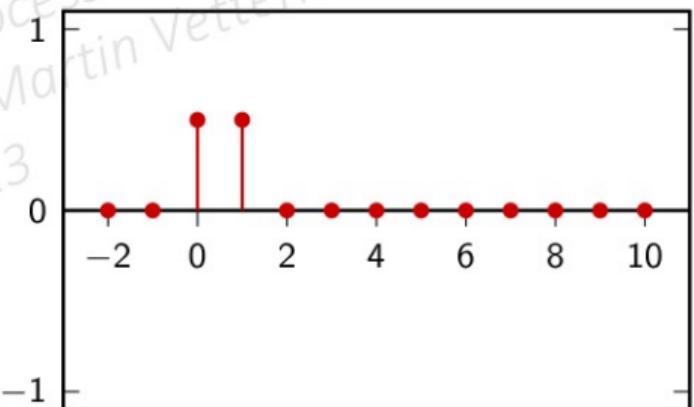
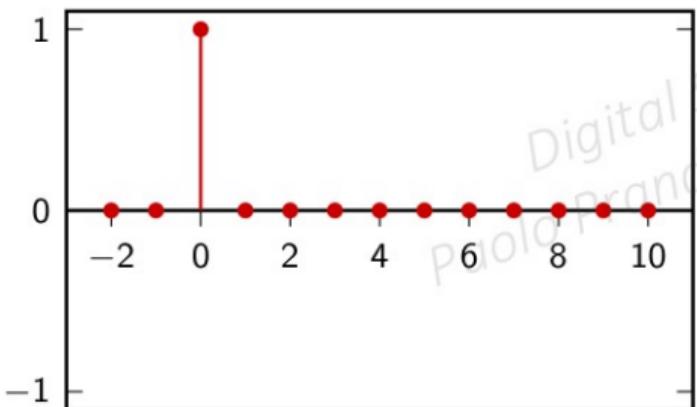
$$y[n] = \frac{x[n] + x[n - 1]}{2}$$

# The 2-point Moving Average Using Lego



Let's average...

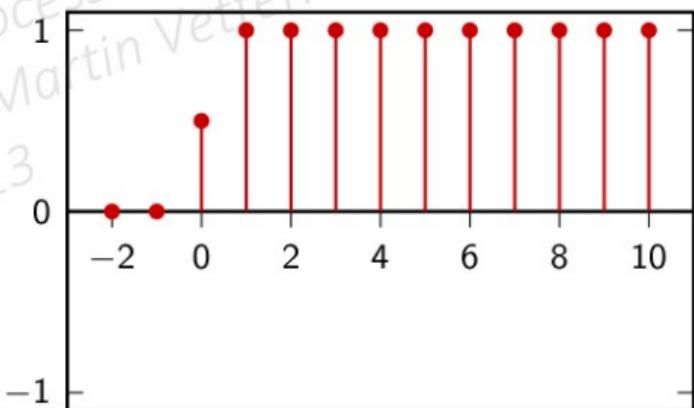
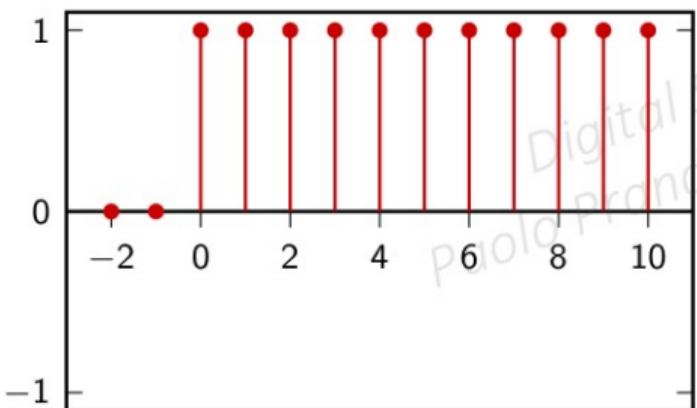
$$x[n] = \delta[n]$$



Digital Signal Processing  
Polo Brandolini and Martin Vetterli  
© 2013

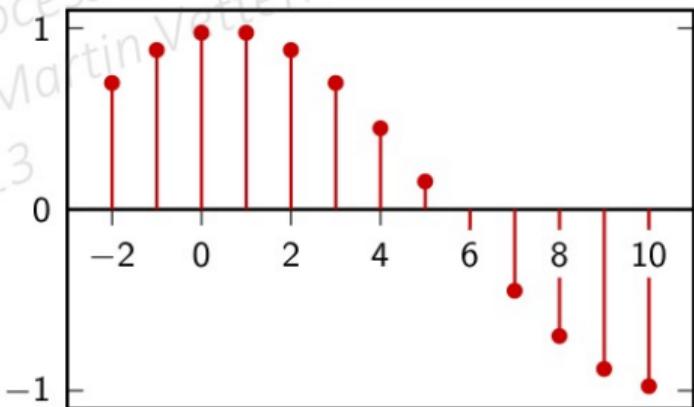
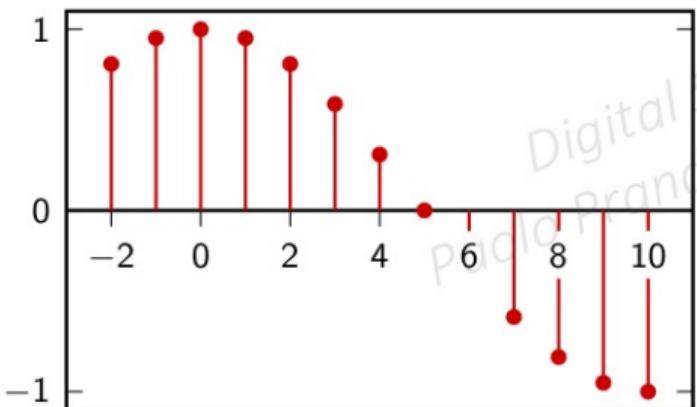
Let's average...

$$x[n] = u[n]$$



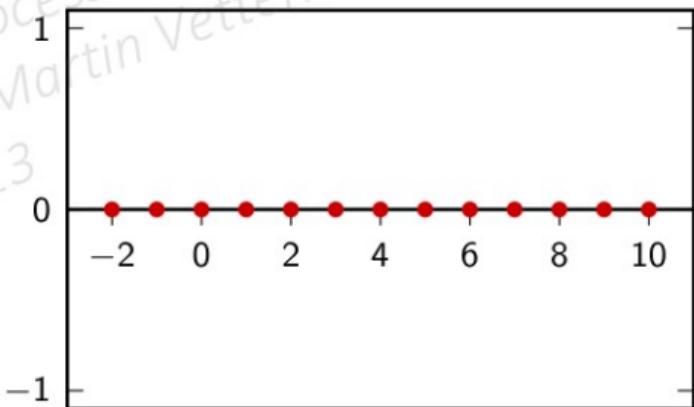
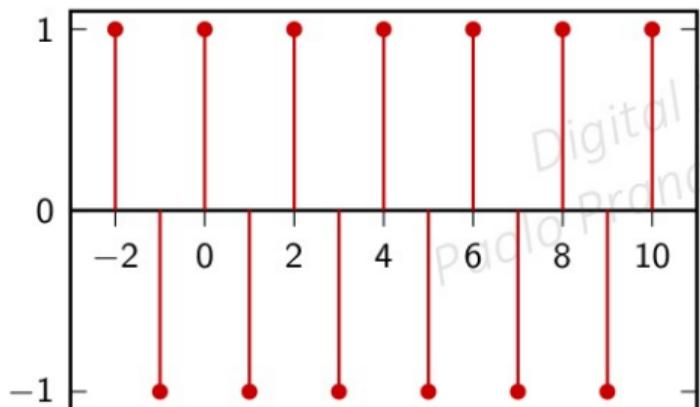
Let's average...

$$x[n] = \cos(\omega n), \quad \omega = \pi/10$$



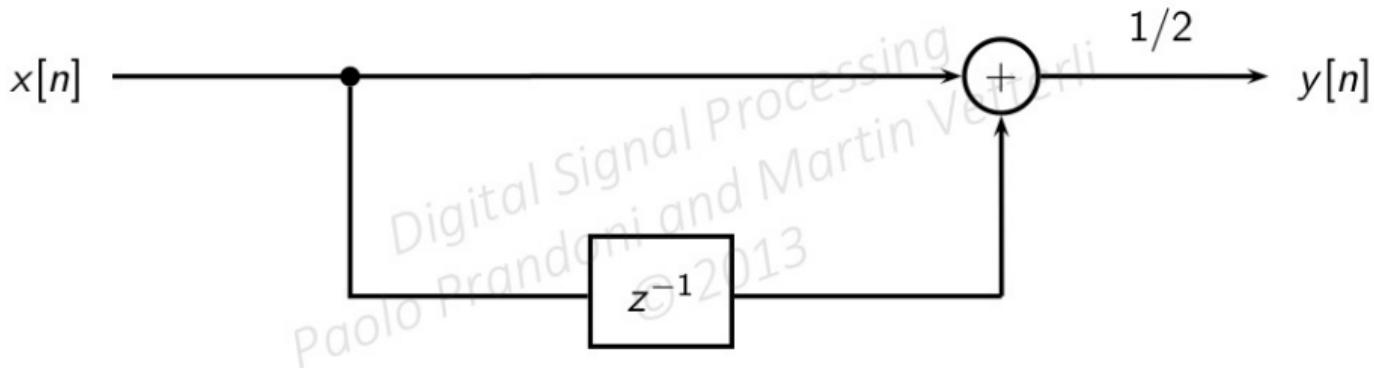
Let's average...

$$x[n] = \cos(\omega n), \quad \omega = \pi$$

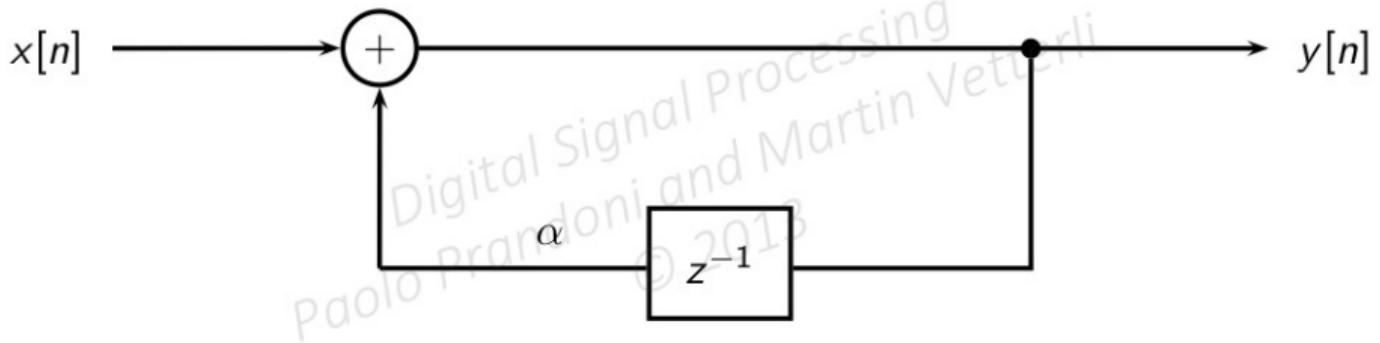


Digital Signal Processing  
Prandoni and Martin Vetterli  
© 2013

# What if we reverse the loop?



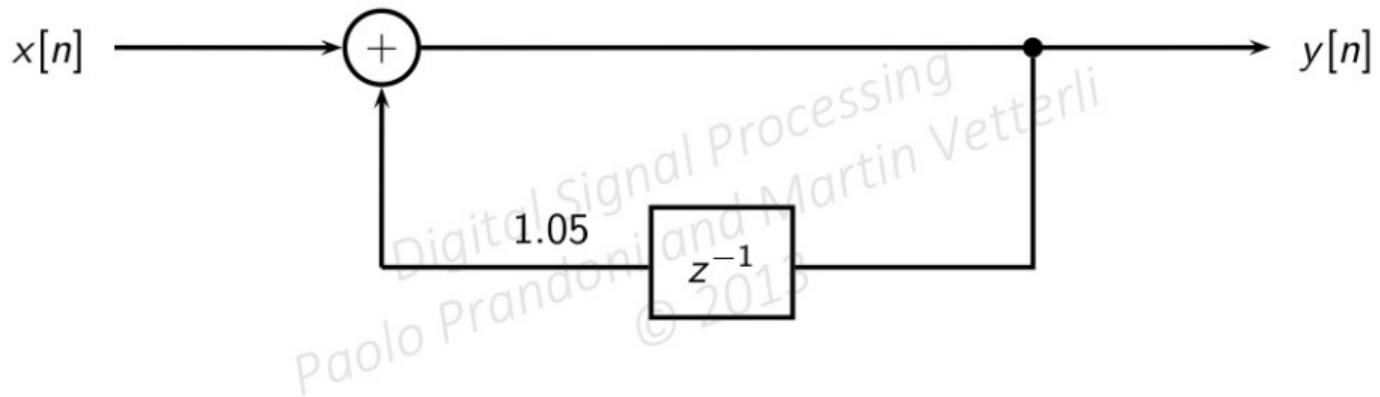
# What if we reverse the loop?



A simple equation to describe compound interest:

- ▶ constant interest/borrowing rate of 5% per year
- ▶ interest accrues on Dec 31
- ▶ deposits/withdrawals during year  $n$ :  $x[n]$
- ▶ balance at year  $n$ :

$$y[n] = 1.05 y[n - 1] + x[n]$$

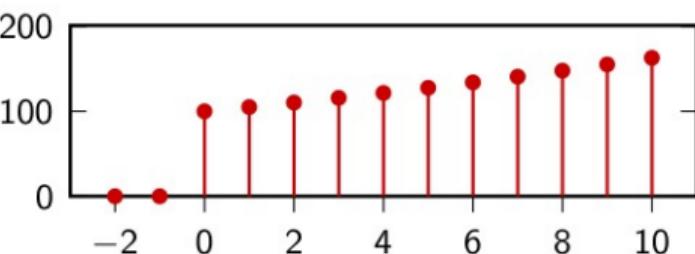
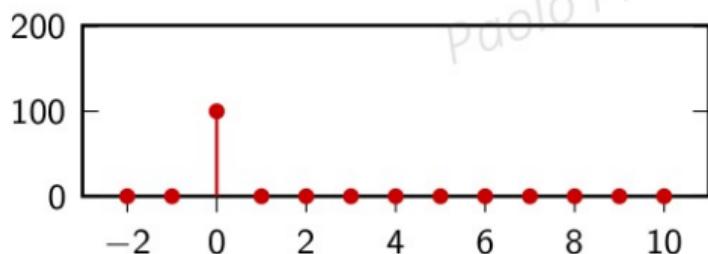


$$y[n] = 1.05 y[n - 1] + x[n]$$

## Example: the one-time investment

$$x[n] = 100 \delta[n]$$

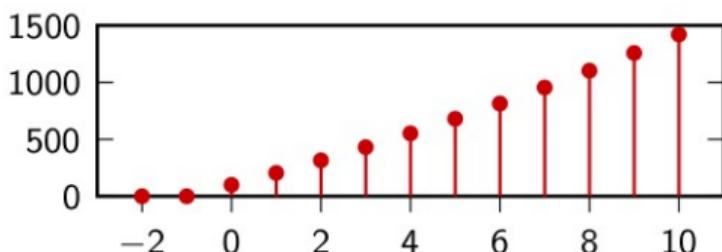
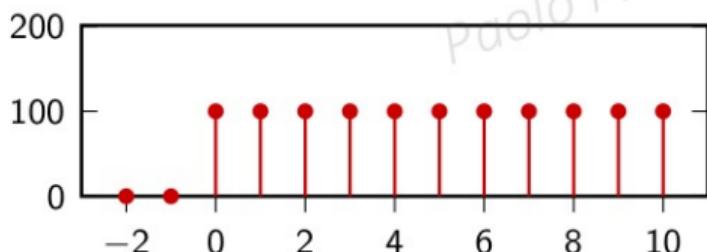
- ▶  $y[0] = 100$
- ▶  $y[1] = 105$
- ▶  $y[2] = 110.25, y[3] = 115.7625$  etc.
- ▶ In general:  $y[n] = (1.05)^n 100 u[n]$



## Example: the saver

$$x[n] = 100 u[n]$$

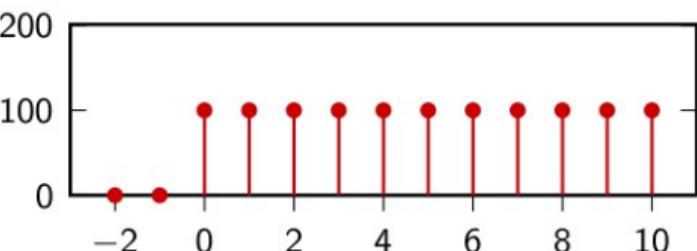
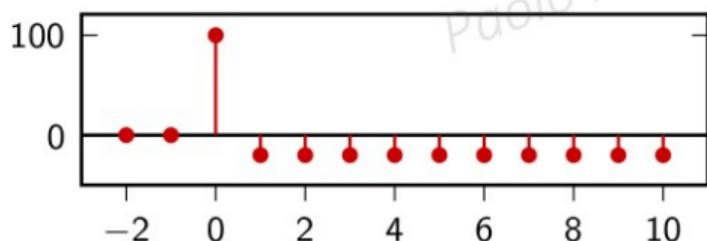
- ▶  $y[0] = 100$
- ▶  $y[1] = 205$
- ▶  $y[2] = 315.25, y[3] = 431.0125$  etc.
- ▶ In general:  $y[n] = 2000 ((1.05)^{n+1} - 1) u[n]$



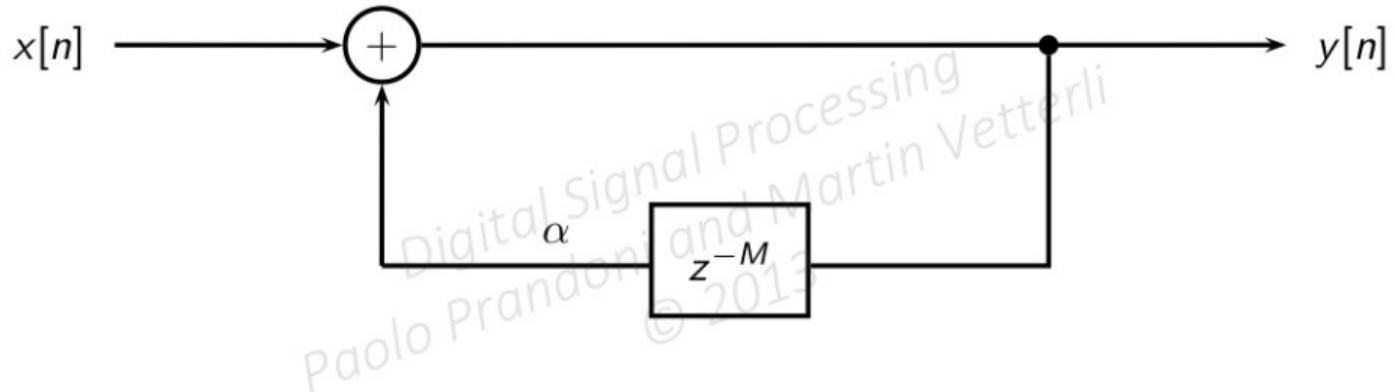
## Example: The independently wealthy

$$x[n] = 100 \delta[n] - 5 u[n - 1]$$

- ▶  $y[0] = 100$
- ▶  $y[1] = 100$
- ▶  $y[2] = 100, y[3] = 100$  etc.
- ▶ In general:  $y[n] = 100 u[n]$



## A simple generalization

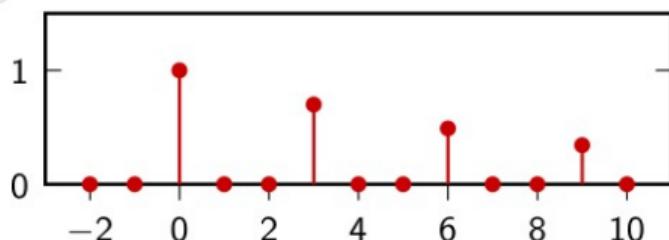
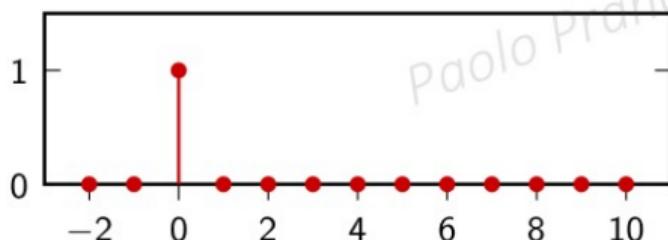


$$y[n] = \alpha y[n - M] + x[n]$$

## Example

$$M = 3, \alpha = 0.7, x[n] = \delta[n]$$

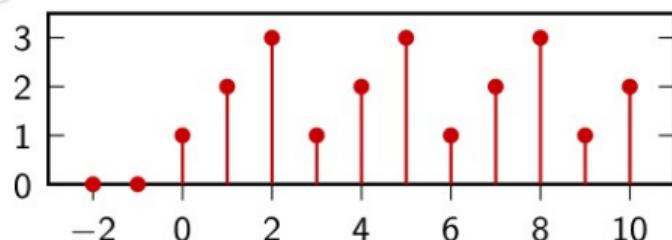
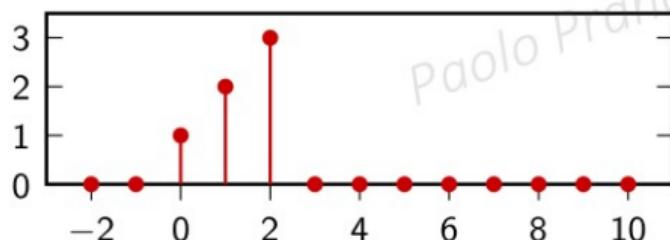
- ▶  $y[0] = 1, y[1] = 0, y[2] = 0$
- ▶  $y[3] = 0.7, y[4] = 0, y[5] = 0$
- ▶  $y[6] = 0.7^2, y[7] = 0, y[8] = 0$ , etc.



## Example

$$M = 3, \alpha = 1, x[n] = \delta[n] + 2\delta[n - 1] + 3\delta[n - 2]$$

- ▶  $y[0] = 1, y[1] = 2, y[2] = 3$
- ▶  $y[3] = 1, y[4] = 2, y[5] = 3$
- ▶  $y[6] = 1, y[7] = 2, y[8] = 3, \text{ etc.}$



- ▶ build a recursion loop with a delay of  $M$
- ▶ choose a signal  $\bar{x}[n]$  that is nonzero only for  $0 \leq n < M$
- ▶ choose a decay factor
- ▶ input  $\bar{x}[n]$  to the system
- ▶ play the output

Digital Signal Processing  
Paolo Prandoni and Martin Etterli  
© 2013

# How do we “play” it, really?

- ▶  $M$ -tap delay →  $M$ -sample “periodicity”
- ▶ associate time  $T$  to sample interval
- ▶ periodic signal of frequency

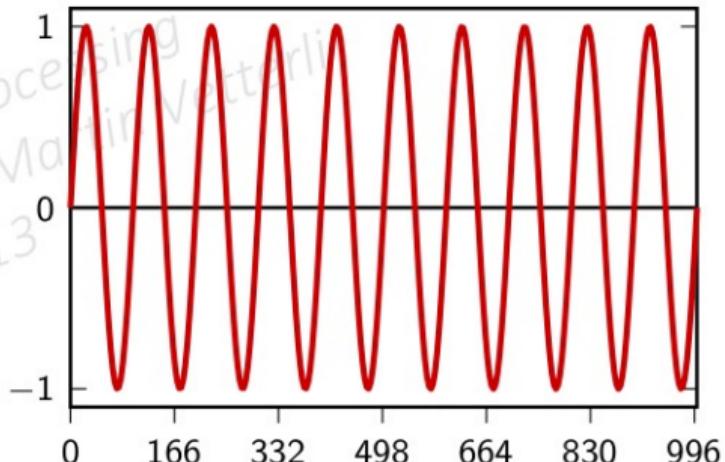
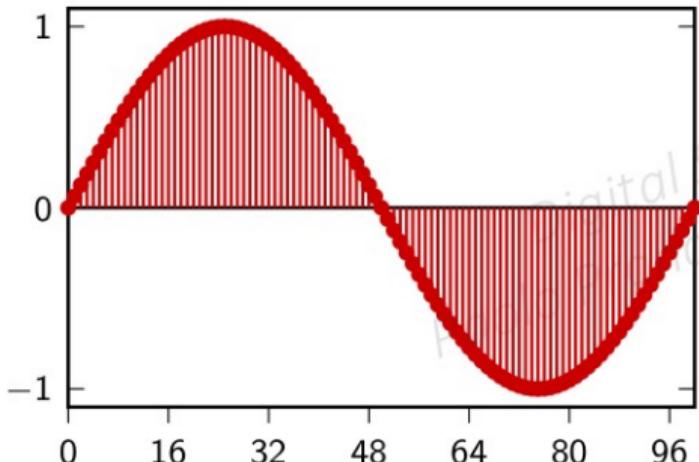
$$f = \frac{1}{MT} \text{Hz}$$

- ▶ example:  $T = 22.7\mu s$ ,  $M = 100$

$$f \approx 440 \text{Hz}$$

# Playing a sine wave

$M = 100, \alpha = 1, \bar{x}[n] = \sin(2\pi n/100)$  for  $0 \leq n < 100$  and zero elsewhere

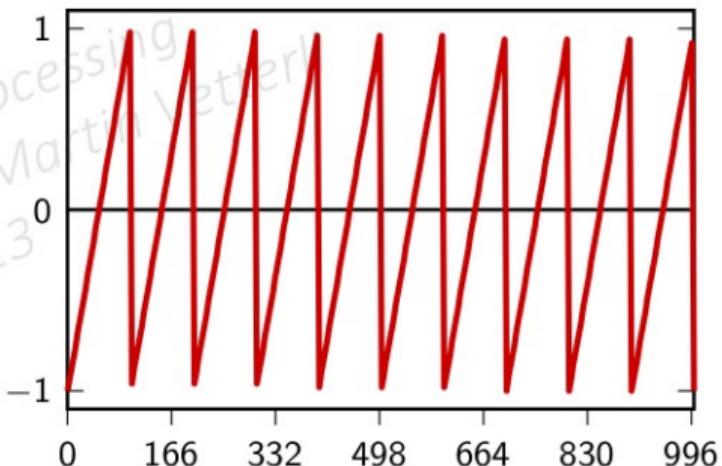
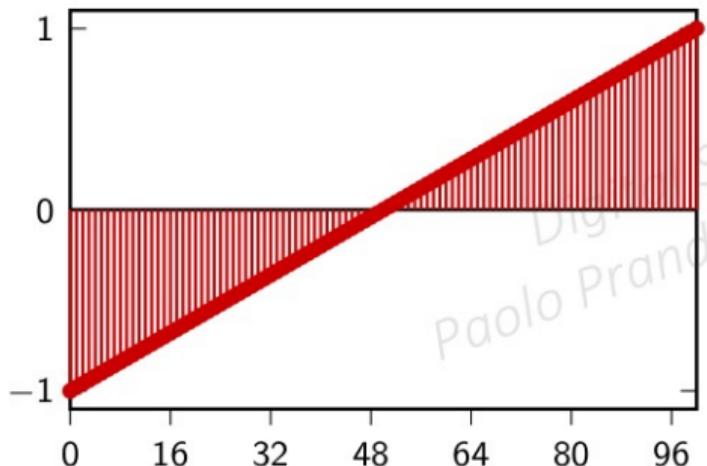


- ▶  $M$  controls frequency (pitch)
- ▶  $\alpha$  controls envelope (decay)
- ▶  $\bar{x}[n]$  controls color (timbre)

Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013

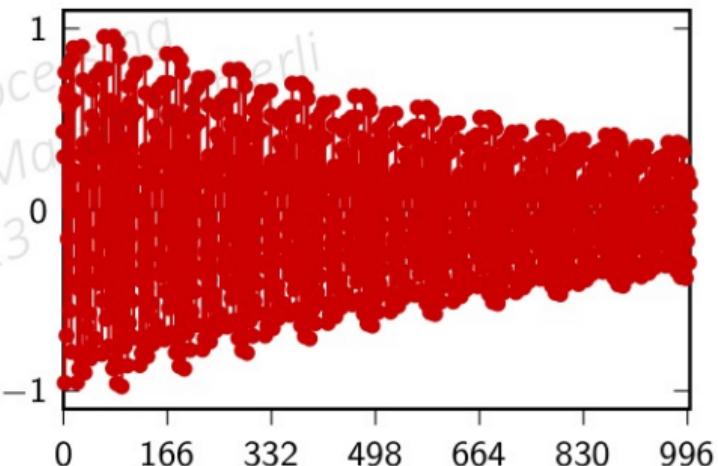
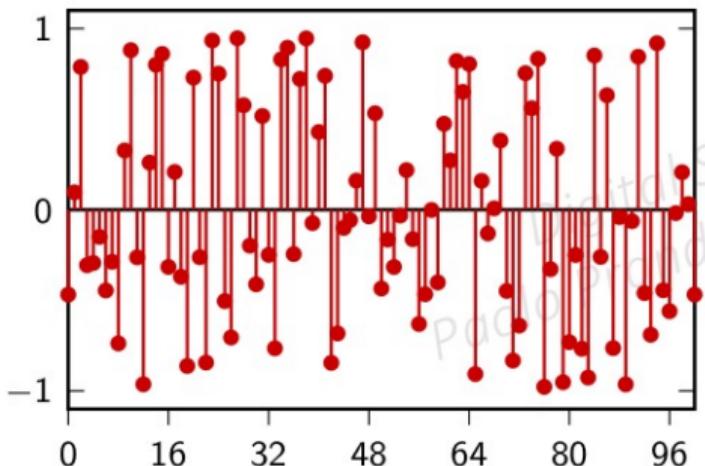
# A proto-violin

$M = 100, \alpha = 0.95, \bar{x}[n]$ : zero-mean sawtooth wave between 0 and 99, zero elsewhere



# The Karplus-Strong Algorithm

$M = 100, \alpha = 0.9, \bar{x}[n]$ : 100 random values between 0 and 99, zero elsewhere



Signal Processing  
and Machine  
Learning  
© 2013

- ▶ We have seen basic elements:
  - adders
  - multipliers
  - delays
- ▶ We have seen two systems
  - moving averages
  - recursive systems
- ▶ We were able to build simple systems with interesting properties
- ▶ to understand all of this in more details we need a mathematical framework!

Digital Signal Processing  
Paolo Prandoni and Martin Vetterli  
© 2013