COM303: Digital Signal Processing

Lecture 2: Discrete-Time Signals

Module Overview:

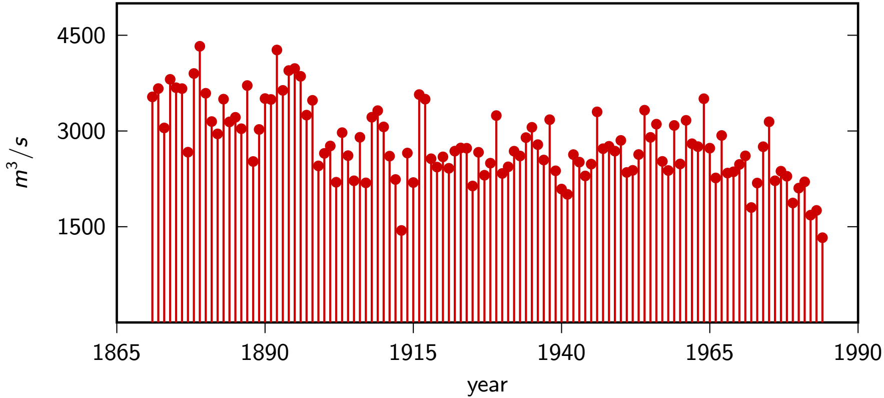
- discrete-time signals
- elementary signal operations
- the Karplus-Strong algorithm

Discrete-time signals have a long tradition...

Meteorology (limnology): the floods of the Nile

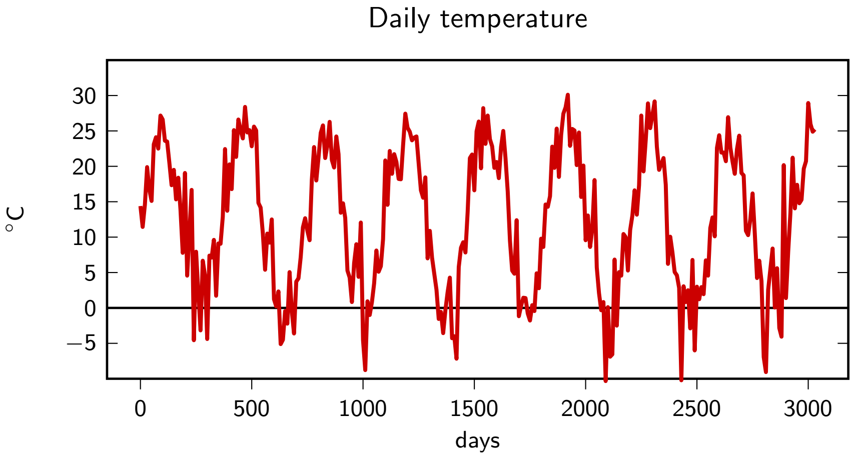
Representations of flood data: circa 2500 BC

Discrete-time signals have a long tradition...

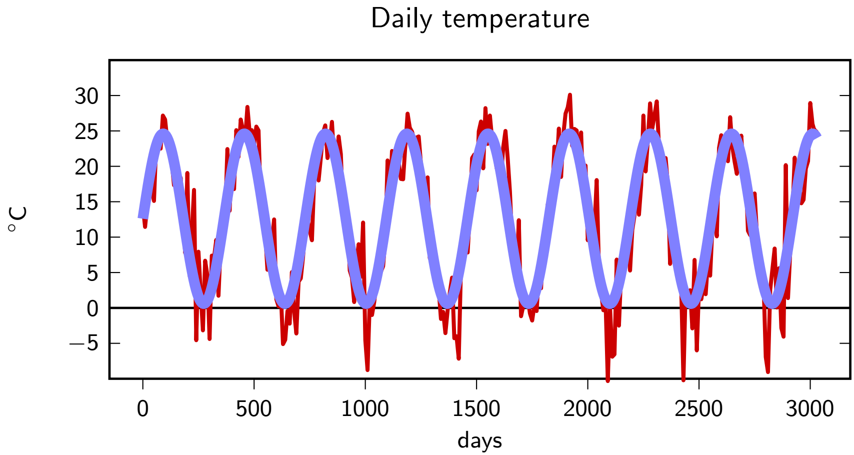


Representations of flood data: circa AD 2000

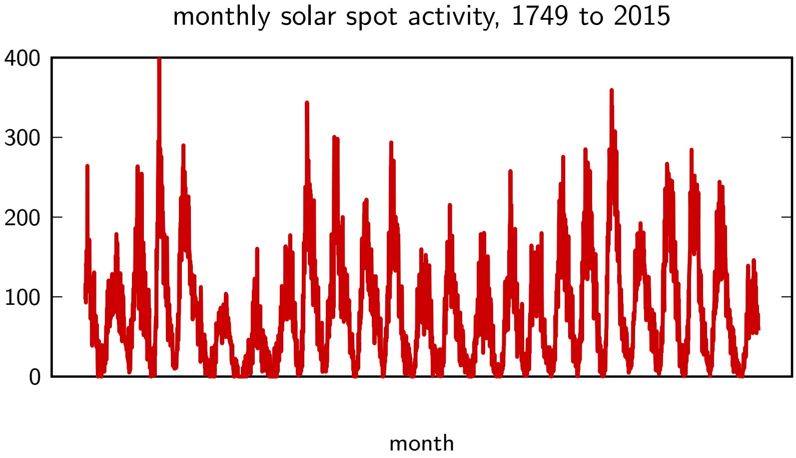
Probably your first scientific experiment...



Probably your first scientific experiment...

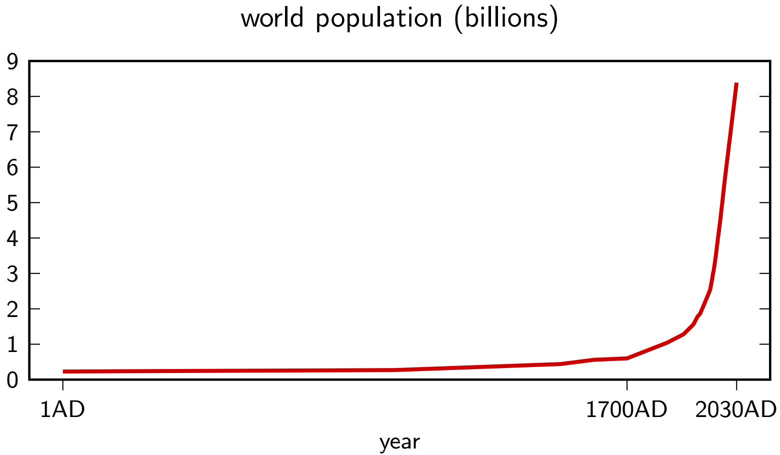


Astronomy



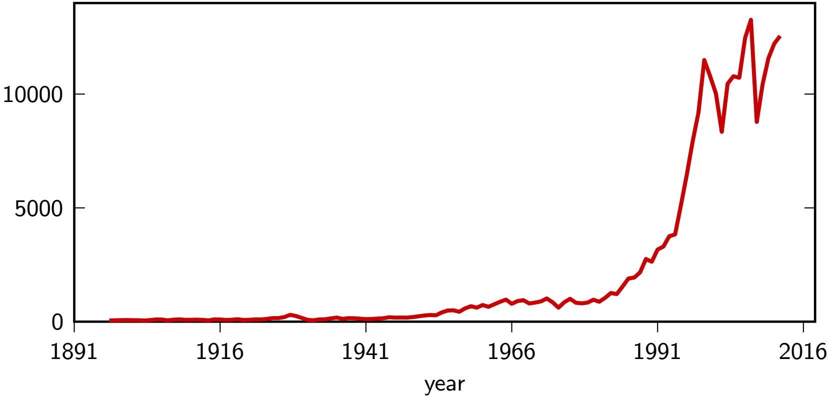
monthly solar spot activity, 1749 to 2015
400
300
200
100
0
month

History and sociology



Economics

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



More formally...

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 *a-dimensional* "time"
- analysis: periodic measurement
- synthesis: stream of generated samples

More formally...

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

- one dimension (for now)
- notation: $x[n]$
- two-sided sequences: $x : \mathbb{Z} \to \mathbb{C}$
- n is *a-dimensional* "time"
- analysis: periodic measurement
- synthesis: stream of generated samples

More formally...

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

- one dimension (for now)
- notation: $x[n]$
- two-sided sequences: $x : \mathbb{Z} \to \mathbb{C}$
- n is *a-dimensional* "time"
- analysis: periodic measurement
- synthesis: stream of generated samples

More formally...

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

- one dimension (for now)
- notation: $x[n]$
- two-sided sequences: $x : \mathbb{Z} \to \mathbb{C}$
- n is *a-dimensional* "time"
- analysis: periodic measurement
- synthesis: stream of generated samples

More formally...

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

- one dimension (for now)
- notation: $x[n]$
- two-sided sequences: $x : \mathbb{Z} \to \mathbb{C}$
- n is *a-dimensional* "time"
- analysis: periodic measurement
- synthesis: stream of generated samples

More formally...

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

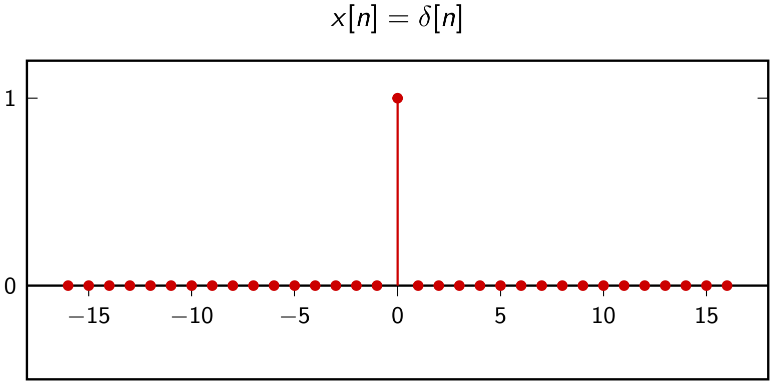
- one dimension (for now)
- notation: $x[n]$
- two-sided sequences: $x : \mathbb{Z} \to \mathbb{C}$
- n is *a-dimensional* "time"
- analysis: periodic measurement
- synthesis: stream of generated samples

More formally...

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

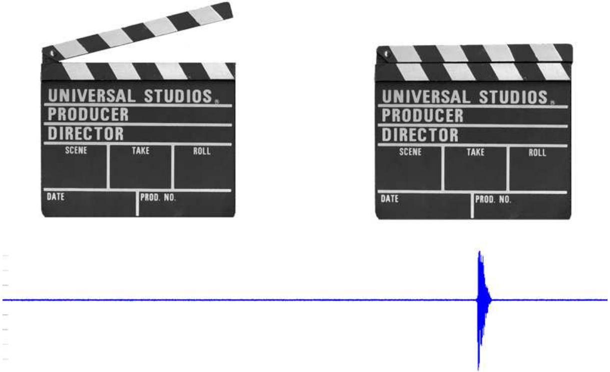
- one dimension (for now)
- notation: $x[n]$
- two-sided sequences: $x : \mathbb{Z} \to \mathbb{C}$
- n is *a-dimensional* “time”
- analysis: periodic measurement
- synthesis: stream of generated samples

The delta signal



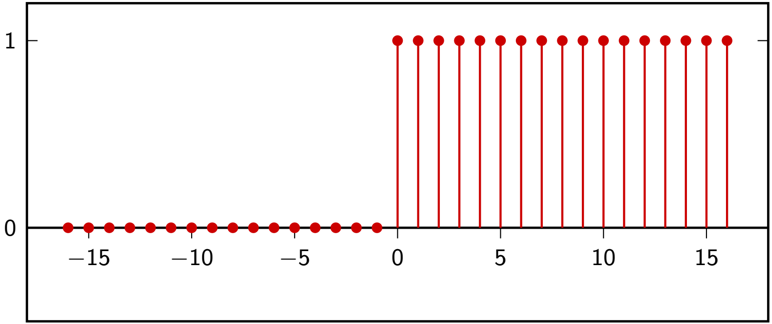
How do you synchronize audio and video...

How do you synchronize audio and video...



The unit step

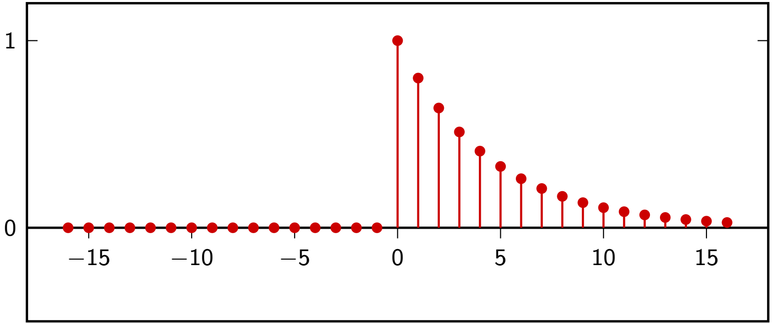
$$x[n] = u[n]$$



The Frankenstein switch...

The exponential decay

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



How fast does your coffee get cold...

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

How fast does your coffee get cold...

Newton's law of cooling:

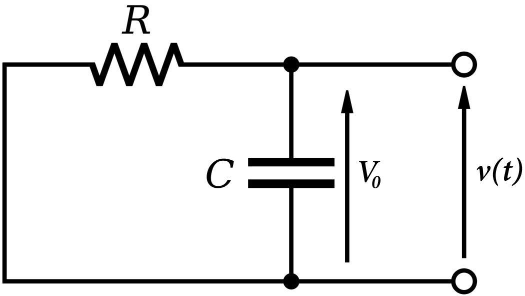
$$\frac{dT}{dt} = -c(T - T_{\text{env}})$$

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

In practice:

- must have convection only
- must have large conductivity

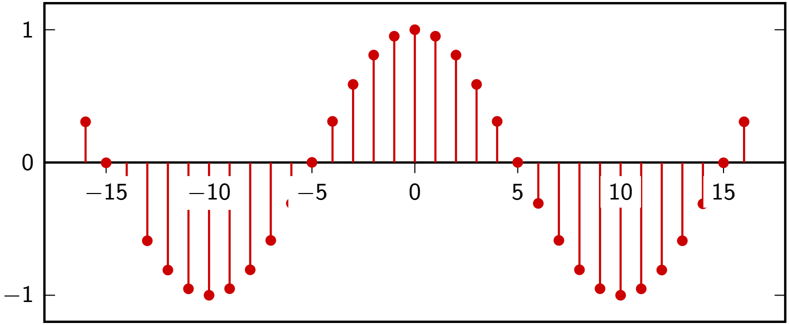
Also, how fast your capacitor discharges



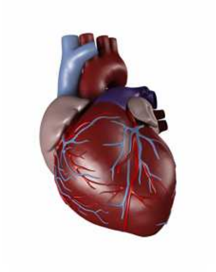
$$v(t) = V_0\, e^{-\frac{t}{RC}}$$

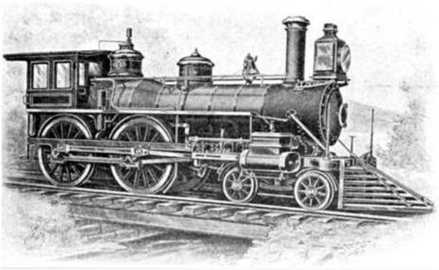
The sinusoid

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



Oscillations are everywhere!





Four signal classes

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

Four signal classes

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

Four signal classes

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

Four signal classes

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

Finite-length signals

- sequence notation: $x[n], \quad n = 0, 1, \ldots, N-1$
- vector notation: $\mathbf{x} = [x_0 \, x_1 \, \ldots \, x_{N-1}]^T$
- practical entities, good for numerical packages (e.g.numpy)

Finite-length signals

- sequence notation: $x[n], \quad n = 0, 1, \ldots, N-1$
- vector notation: $\mathbf{x} = [x_0 \, x_1 \ldots x_{N-1}]^T$
- practical entities, good for numerical packages (e.g.numpy)

Finite-length signals

- sequence notation: $x[n], \quad n = 0, 1, \ldots, N-1$
- vector notation: $\mathbf{x} = [x_0\, x_1 \ldots x_{N-1}]^T$
- practical entities, good for numerical packages (e.g.numpy)

Infinite-length signals

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

Infinite-length signals

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

Periodic signals

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

Periodic signals

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

Periodic signals

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

Finite-support signals

- Finite-support sequence:

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

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

Finite-support signals

- Finite-support sequence:

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

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

Finite-support signals

- Finite-support sequence:

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

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

Elementary operators

- 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):

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

Elementary operators

- 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):

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

Elementary operators

- 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):

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

Elementary operators

- scaling:

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

- sum:

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

- product:

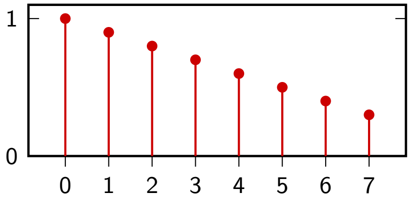
$$y[n] = x[n] \cdot z[n]$$

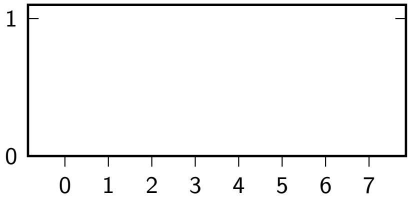
- shift by k (delay):

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

Shift of a finite-length: finite-support

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

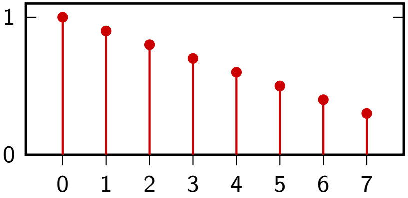


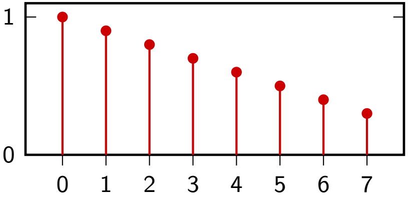


Shift of a finite-length: finite-support

$$x[n]$$

$$\ldots \; \boxed{x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7} \; \ldots$$

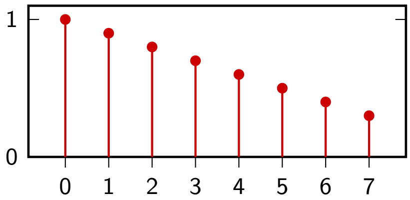


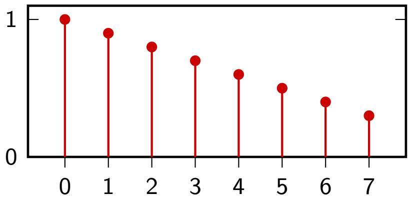


Shift of a finite-length: finite-support

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

$$\ldots \quad 0 \quad 0 \quad 0 \quad \boxed{x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7} \quad 0 \quad 0 \quad 0 \quad \ldots$$





Shift of a finite-length: finite-support

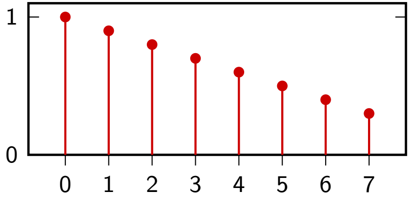
$$\bar{x}[n-1]$$

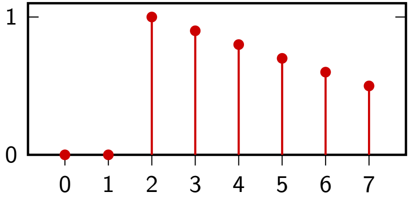
$$\ldots \quad 0 \quad 0 \quad 0 \quad \boxed{0 \quad x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6} \quad x_7 \quad 0 \quad 0 \quad \ldots$$

Shift of a finite-length: finite-support

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

$$\ldots \quad 0 \quad 0 \quad 0 \quad \boxed{0 \quad 0 \quad x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5} \quad x_6 \quad x_7 \quad 0 \quad \ldots$$

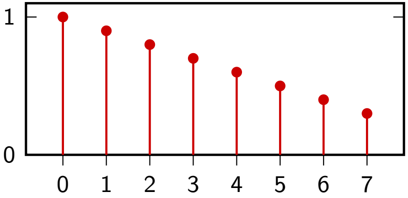


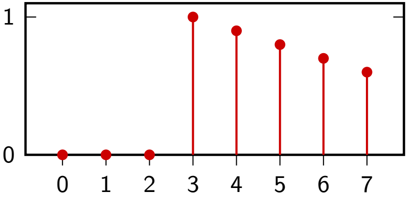


Shift of a finite-length: finite-support

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

$$\ldots \quad 0 \quad 0 \quad 0 \quad \boxed{0 \quad 0 \quad 0 \quad x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4} \quad x_5 \quad x_6 \quad x_7 \quad \ldots$$





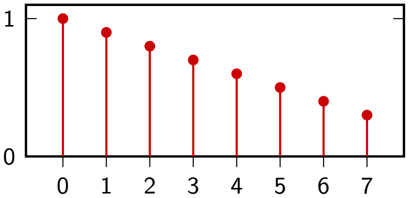
Shift of a finite-length: finite-support

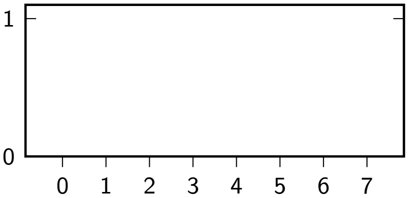
$$\bar{x}[n-4]$$

$$\ldots \quad 0 \quad 0 \quad 0 \quad \boxed{0 \quad 0 \quad 0 \quad 0 \quad x_0 \quad x_1 \quad x_2 \quad x_3} \quad x_4 \quad x_5 \quad x_6 \quad \ldots$$

Shift of a finite-length: periodic extension

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

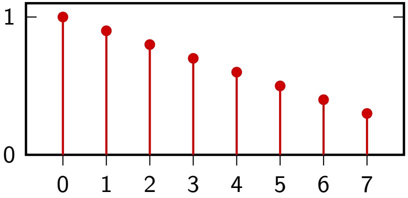


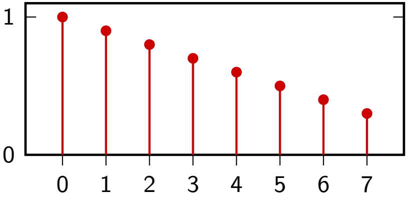


Shift of a finite-length: periodic extension

$$x[n]$$

$$\ldots \boxed{x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7} \ldots$$

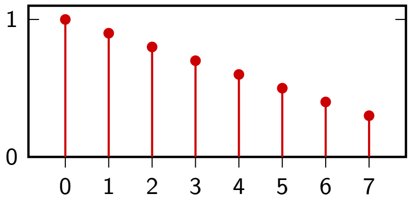


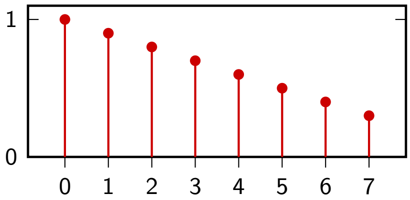


Shift of a finite-length: periodic extension

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

$$\ldots \quad x_5 \quad x_6 \quad x_7 \quad \boxed{x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7} \quad x_0 \quad x_1 \quad x_2 \quad \ldots$$

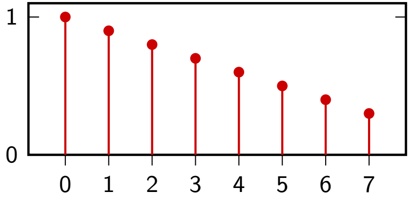


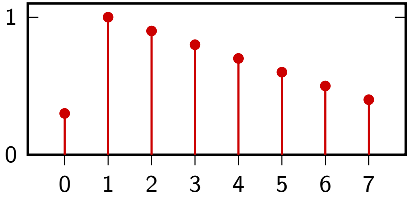


Shift of a finite-length: periodic extension

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

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

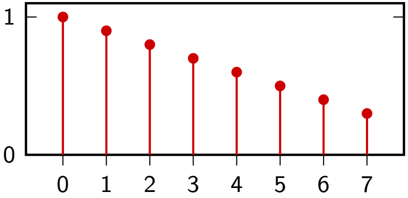


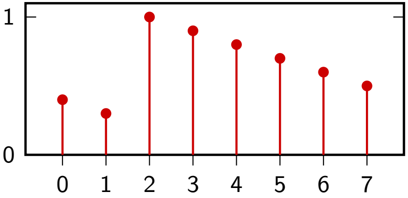


Shift of a finite-length: periodic extension

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

$$\ldots \quad x_3 \quad x_4 \quad x_5 \quad \boxed{x_6 \quad x_7 \quad x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5} \quad x_6 \quad x_7 \quad x_0 \quad \ldots$$





Shift of a finite-length: periodic extension

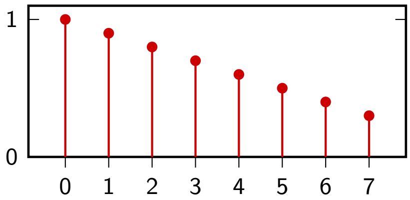
$$\tilde{x}[n-3]$$

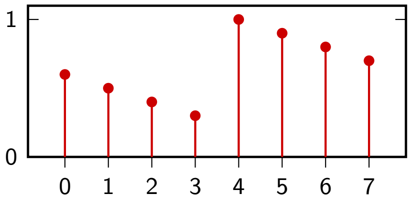
$$\ldots \; x_2 \; x_3 \; x_4 \; \boxed{x_5 \; x_6 \; x_7 \; x_0 \; x_1 \; x_2 \; x_3 \; x_4} \; x_5 \; x_6 \; x_7 \; \ldots$$

Shift of a finite-length: periodic extension

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

$$\ldots \quad x_1 \quad x_2 \quad x_3 \quad \boxed{x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_0 \quad x_1 \quad x_2 \quad x_3} \quad x_4 \quad x_5 \quad x_6 \quad \ldots$$





Energy and power

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

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

Energy and power

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

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

Energy and power: periodic signals

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

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

Energy and power: periodic signals

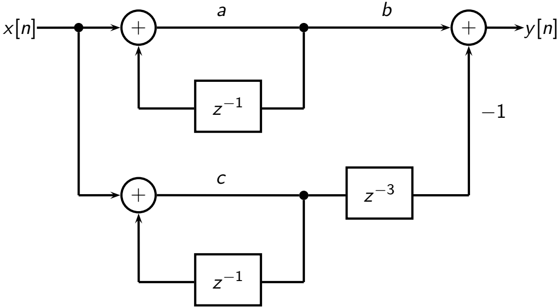
$$E_{\tilde{x}} = \infty$$

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

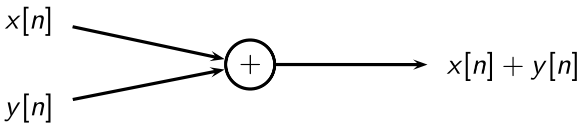
Overview:

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

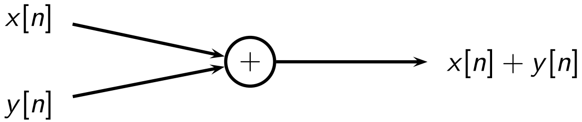
DSP as Lego

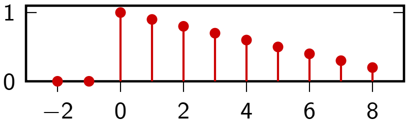


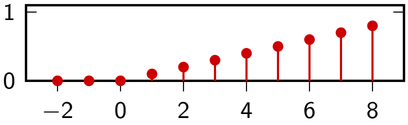
Building Blocks: Adder



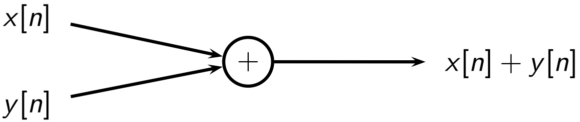
Building Blocks: Adder

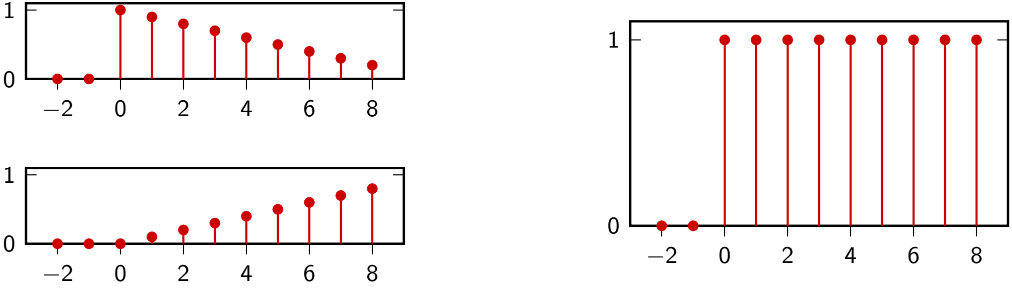






Building Blocks: Adder



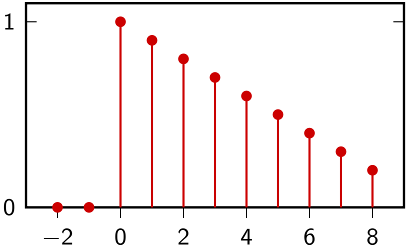


Building Blocks: Multiplier

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

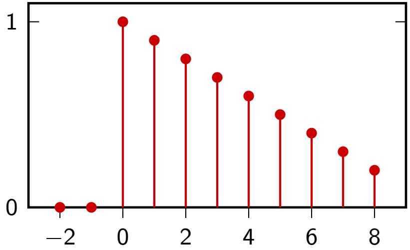
Building Blocks: Multiplier

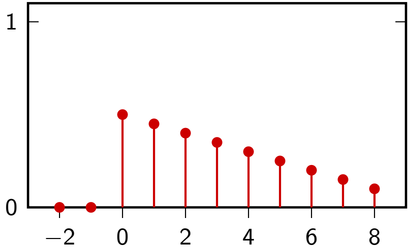
$$x[n] \xrightarrow{\alpha} \alpha x[n]$$



Building Blocks: Multiplier

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



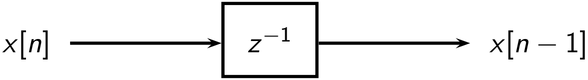


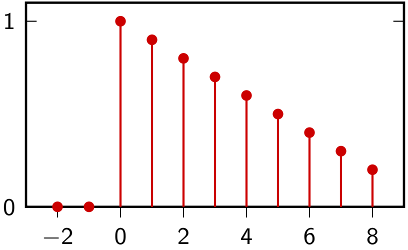
$\alpha = 0.5$

Building Blocks: Unit Delay

$$x[n] \longrightarrow \boxed{z^{-1}} \longrightarrow x[n-1]$$

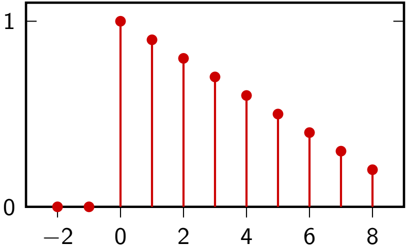
Building Blocks: Unit Delay

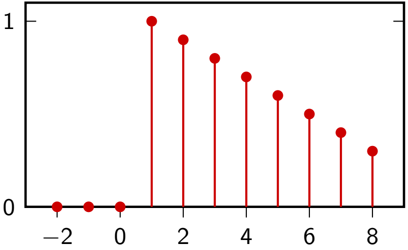




Building Blocks: Unit Delay

$$x[n] \longrightarrow \boxed{z^{-1}} \longrightarrow x[n-1]$$



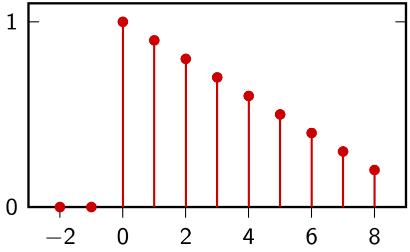


Building Blocks: Arbitrary Delay

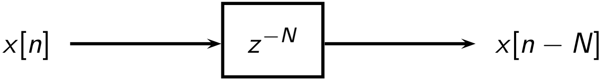
$$x[n] \longrightarrow \boxed{z^{-N}} \longrightarrow x[n-N]$$

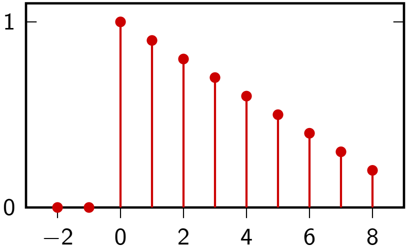
Building Blocks: Arbitrary Delay

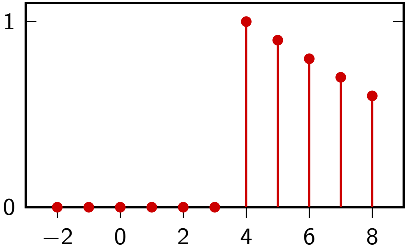
$$x[n] \longrightarrow \boxed{z^{-N}} \longrightarrow x[n-N]$$



Building Blocks: Arbitrary Delay







$N = 4$

The 2-point Moving Average

- 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

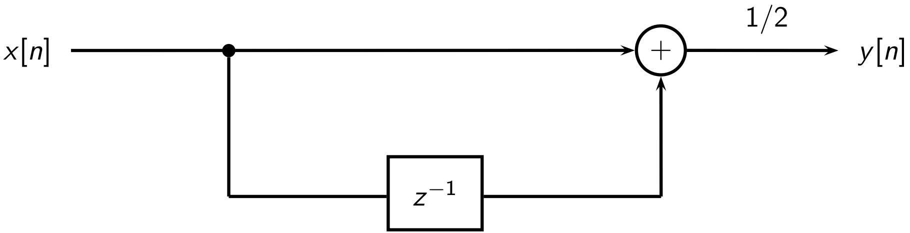
- simple average:

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

- moving average: take a “local” average

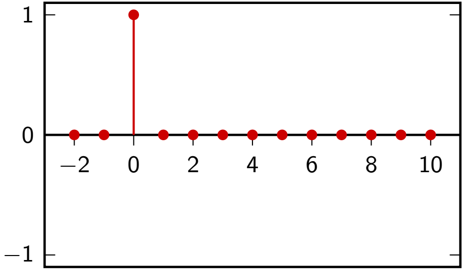
$$y[n] = \frac{x[n] + x[n-1]}{2}$$

The 2-point Moving Average Using Lego



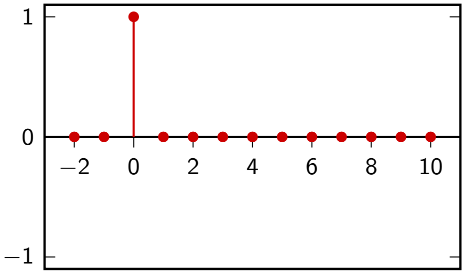
Let's average...

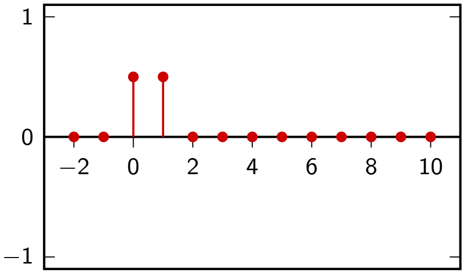
$x[n] = \delta[n]$



Let's average...

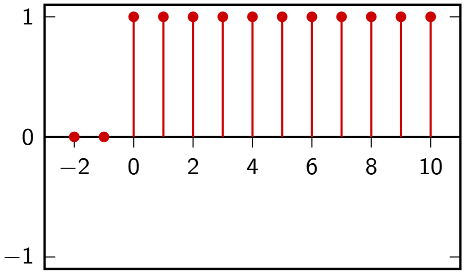
$x[n] = \delta[n]$





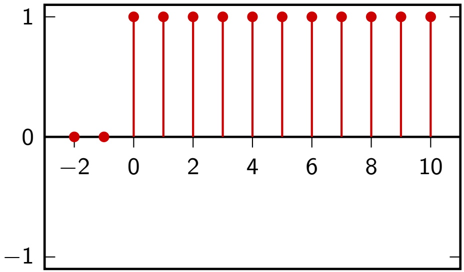
Let's average...

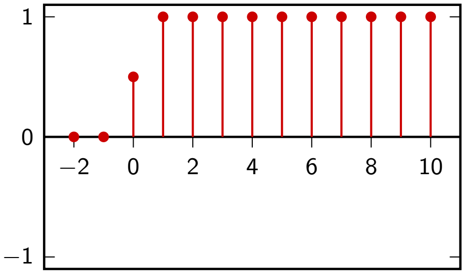
$x[n] = u[n]$



Let's average...

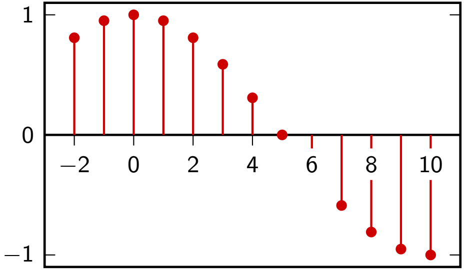
$x[n] = u[n]$





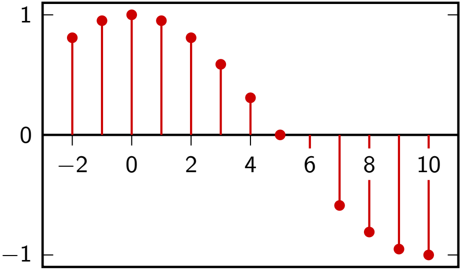
Let's average...

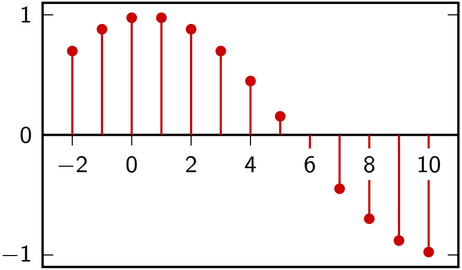
$$x[n] = \cos(\omega n), \quad \omega = \pi/10$$



Let's average...

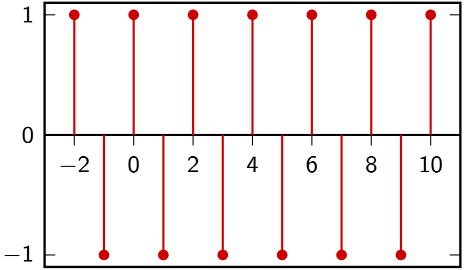
$$x[n] = \cos(\omega n), \quad \omega = \pi/10$$





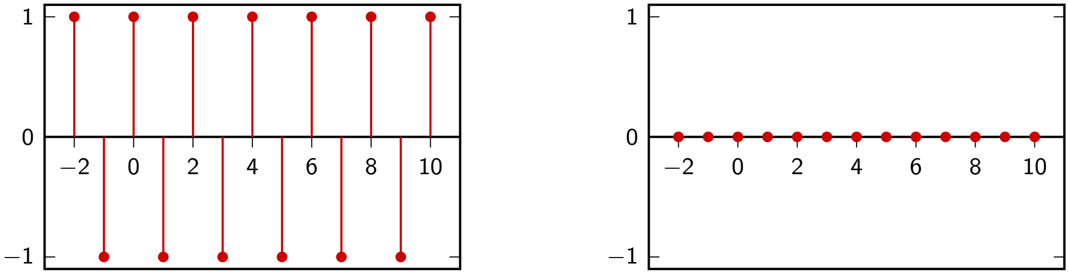
Let's average...

$x[n] = (-1)^n$

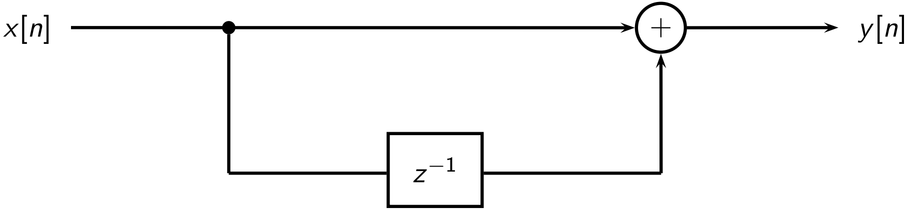


Let's average...

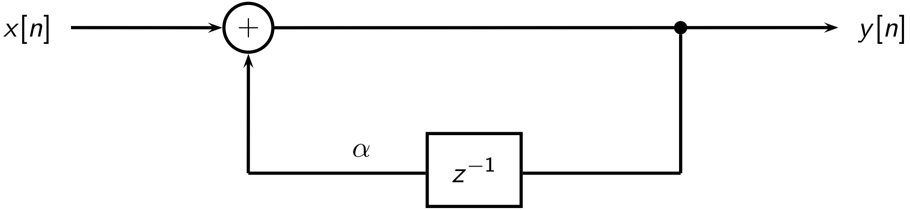
$x[n] = (-1)^n$



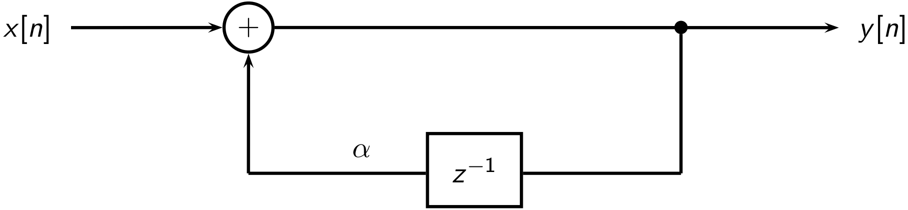
What if we reverse the loop?



What if we reverse the loop?

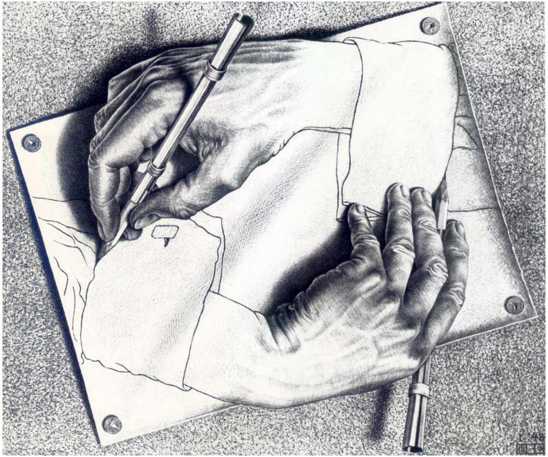


What if we reverse the loop?



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

A powerful concept: recursion



How we solve the chicken-and-egg problem

Zero Initial Conditions

- set a start time (usually $n_0 = 0$)
- assume input and output are zero for *all time* before n_0

A simple model for banking

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:

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

A simple model for banking

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:

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

A simple model for banking

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:

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

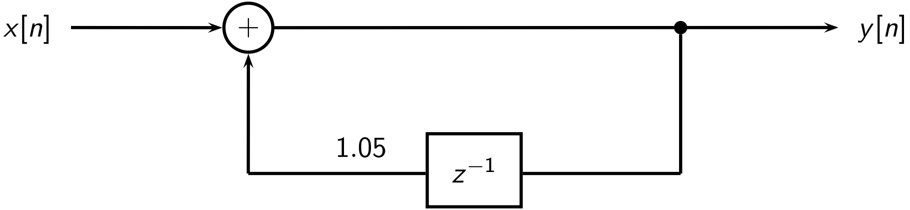
A simple model for banking

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:

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

Accumulation of interest: first-order recursion

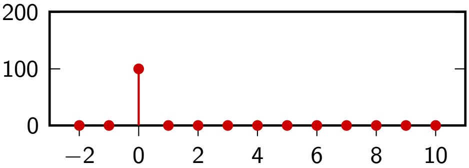


$$y[n] = 1.05\, y[n-1] + x[n]$$

Example: the one-time investment

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

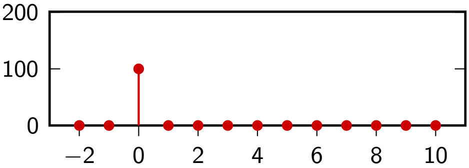
- $y[0] = 100$
- $y[1] = 105$
- $y[2] = 110.25$, $y[3] = 115.7625$ etc.
- In general: $y[n] = (1.05)^n 100\, u[n]$



Example: the one-time investment

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

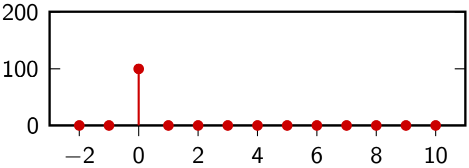
- $y[0] = 100$
- $y[1] = 105$
- $y[2] = 110.25$, $y[3] = 115.7625$ etc.
- In general: $y[n] = (1.05)^n 100\, u[n]$



Example: the one-time investment

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

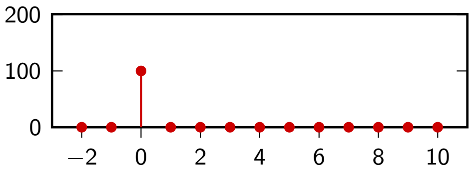
- $y[0] = 100$
- $y[1] = 105$
- $y[2] = 110.25$, $y[3] = 115.7625$ etc.
- In general: $y[n] = (1.05)^n 100\, u[n]$



Example: the one-time investment

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

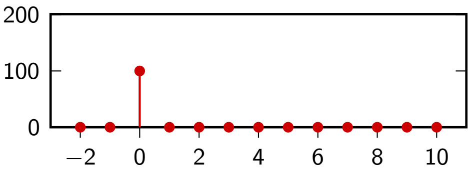
- $y[0] = 100$
- $y[1] = 105$
- $y[2] = 110.25, \; y[3] = 115.7625$ etc.
- In general: $y[n] = (1.05)^n 100\, u[n]$

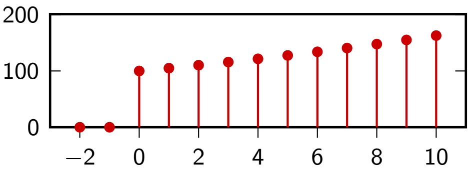


Example: the one-time investment

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

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

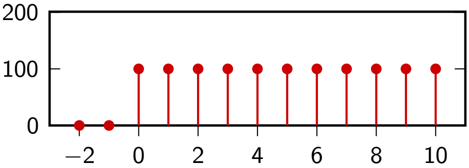




Example: the saver

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

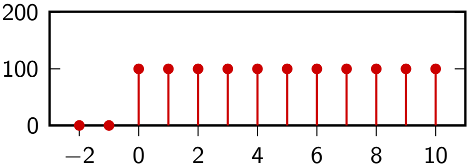
- $y[0] = 100$
- $y[1] = 205$
- $y[2] = 315.25,\ y[3] = 431.0125$ etc.
- In general: $y[n] = 2000\left((1.05)^{n+1} - 1\right) u[n]$



Example: the saver

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

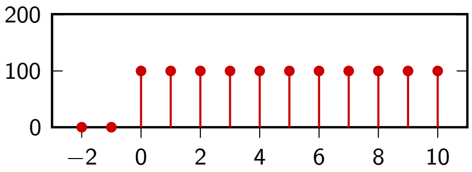
- $y[0] = 100$
- $y[1] = 205$
- $y[2] = 315.25$, $y[3] = 431.0125$ etc.
- In general: $y[n] = 2000\,((1.05)^{n+1} - 1)\,u[n]$



Example: the saver

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

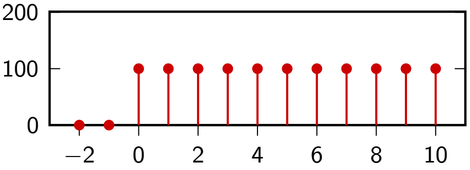
- $y[0] = 100$
- $y[1] = 205$
- $y[2] = 315.25, \ y[3] = 431.0125$ etc.
- In general: $y[n] = 2000\left((1.05)^{n+1} - 1\right) u[n]$



Example: the saver

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

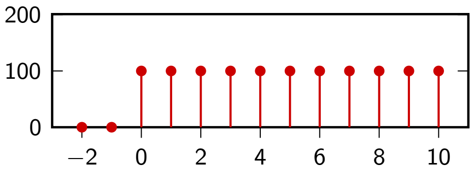
- $y[0] = 100$
- $y[1] = 205$
- $y[2] = 315.25,\ y[3] = 431.0125$ etc.
- In general: $y[n] = 2000\left((1.05)^{n+1} - 1\right) u[n]$

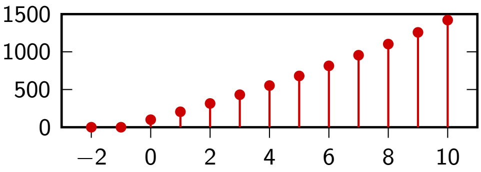


Example: the saver

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

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

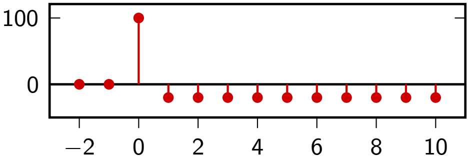




Example: living off the interest

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

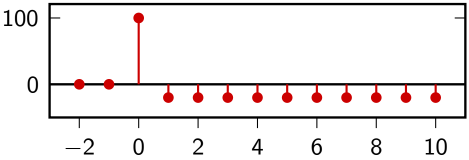
- $y[0] = 100$
- $y[1] = 100$
- $y[2] = 100$, $y[3] = 100$ etc.
- In general: $y[n] = 100\,u[n]$



Example: living off the interest

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

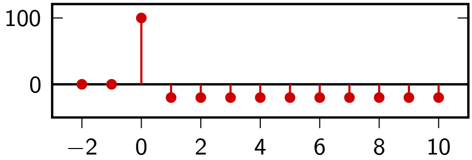
- $y[0] = 100$
- $y[1] = 100$
- $y[2] = 100$, $y[3] = 100$ etc.
- In general: $y[n] = 100\,u[n]$



Example: living off the interest

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

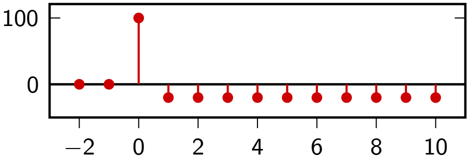
- $y[0] = 100$
- $y[1] = 100$
- $y[2] = 100$, $y[3] = 100$ etc.
- In general: $y[n] = 100\,u[n]$



Example: living off the interest

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

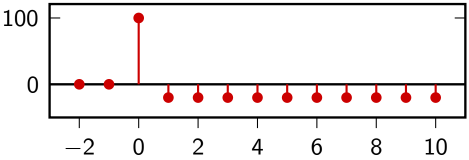
- $y[0] = 100$
- $y[1] = 100$
- $y[2] = 100,\ y[3] = 100$ etc.
- In general: $y[n] = 100\,u[n]$

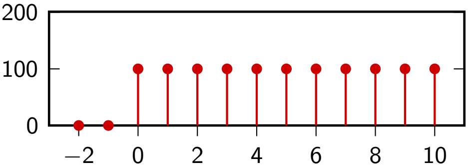


Example: living off the interest

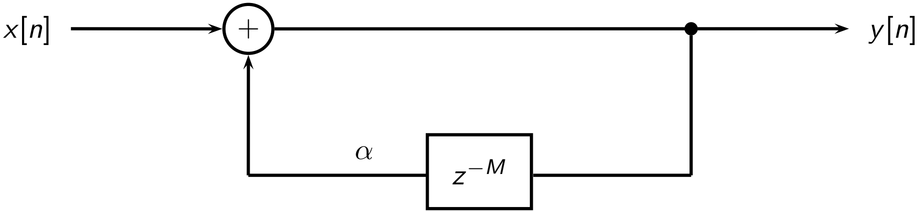
$x[n] = 100\,\delta[n] - 5\,u[n-1]$

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



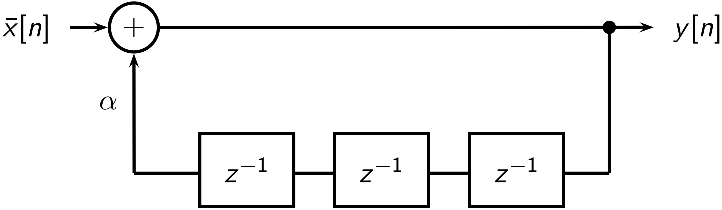


An interesting generalization



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

Creating loops

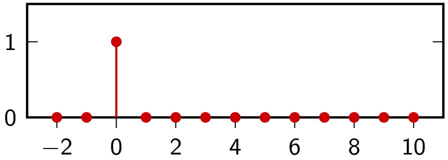


$$y[n] = \alpha y[n-3] + \bar{x}[n]$$

Example

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

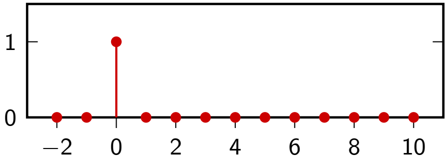
- $y[0] = 1,\ y[1] = 0,\ y[2] = 0$
- $y[3] = 0.7,\ y[4] = 0,\ y[5] = 0$
- $y[6] = 0.7^2,\ y[7] = 0,\ y[8] = 0$, etc.



Example

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

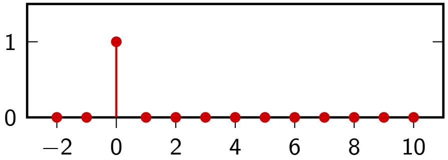
- $y[0] = 1,\ y[1] = 0,\ y[2] = 0$
- $y[3] = 0.7,\ y[4] = 0,\ y[5] = 0$
- $y[6] = 0.7^2,\ y[7] = 0,\ y[8] = 0$, etc.



Example

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

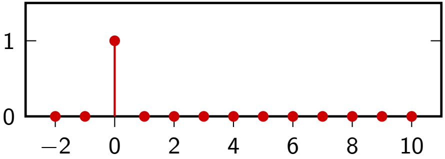
- $y[0] = 1,\ y[1] = 0,\ y[2] = 0$
- $y[3] = 0.7,\ y[4] = 0,\ y[5] = 0$
- $y[6] = 0.7^2,\ y[7] = 0,\ y[8] = 0$, etc.

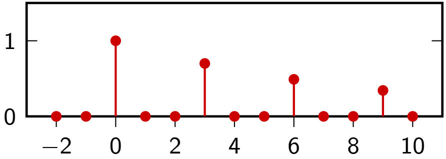


Example

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

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

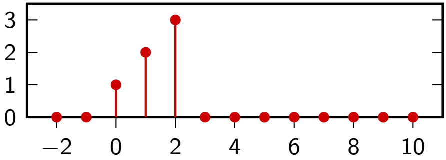


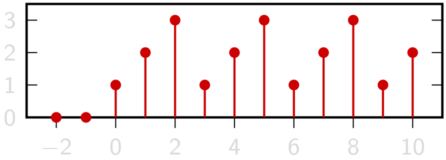


Example

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

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

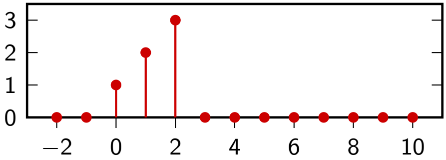


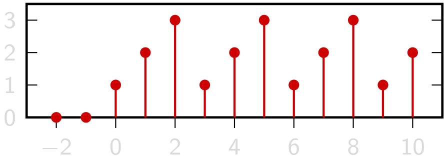


Example

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

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

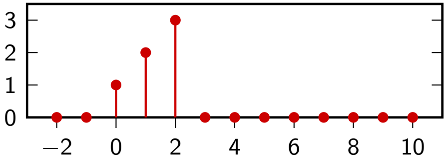


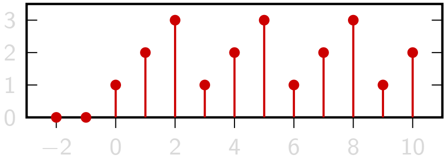


Example

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

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

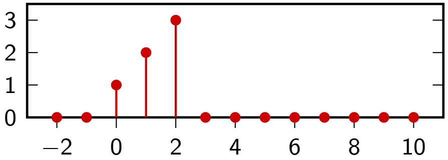


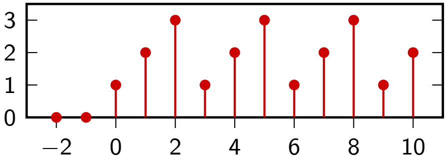


Example

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

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





We can make music with that!

- 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

We can make music with that!

- 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

We can make music with that!

- 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

We can make music with that!

- 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

We can make music with that!

- 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

How do we “play” it, really?

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

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

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

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

How do we “play” it, really?

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

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

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

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

How do we "play" it, really?

- M-tap delay $\longrightarrow$ M-sample "periodicity"
- associate time T to sample interval
- periodic signal of frequency

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

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

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

How do we "play" it, really?

- M-tap delay $\longrightarrow$ M-sample "periodicity"
- associate time T to sample interval
- periodic signal of frequency

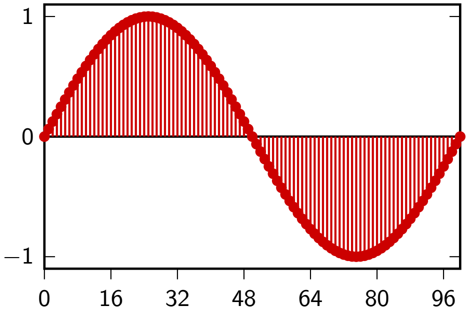
$$f = \frac{1}{MT}\text{Hz}$$

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

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

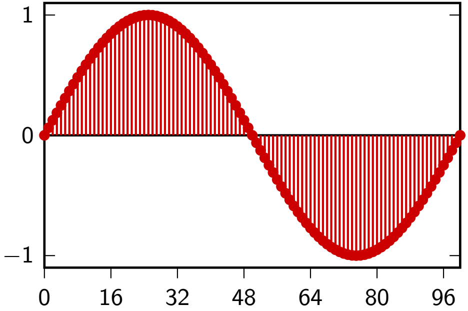
Playing a sine wave

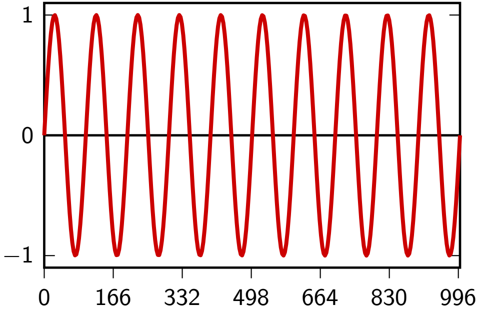
$M = 100$, $\alpha = 1$, $\bar{x}[n] = \sin(2\pi\, n/100)$ for $0 \leq n < 100$ and zero elsewhere



Playing a sine wave

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





Play

Introducing some realism

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

Introducing some realism

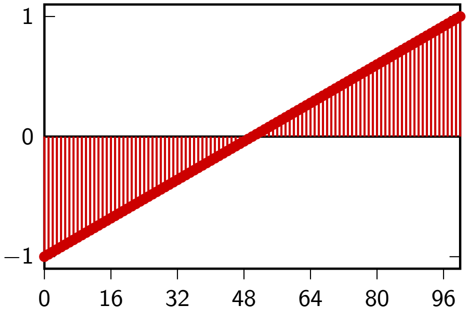
- M controls frequency (pitch)
- α controls envelope (decay)
- $\bar{x}[n]$ controls color (timbre)

Introducing some realism

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

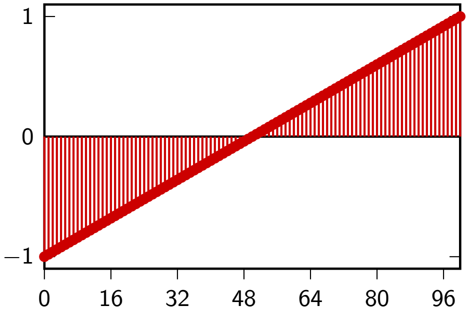
A proto-violin

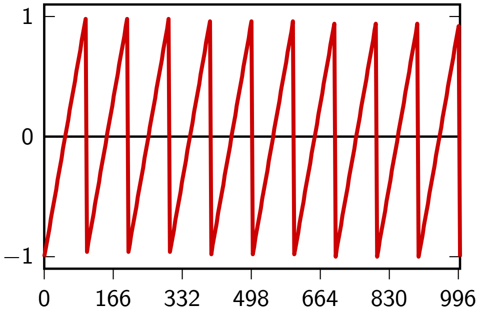
$M = 100$, $\alpha = 0.95$, $\bar{x}[n]$: zero-mean sawtooth wave between 0 and 99, zero elsewhere



A proto-violin

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

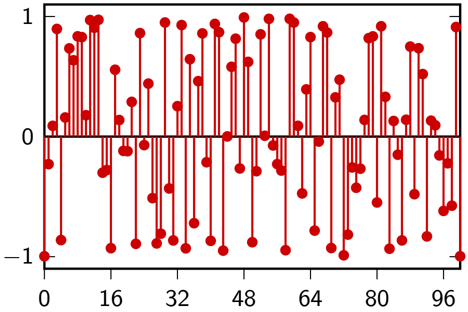




Play

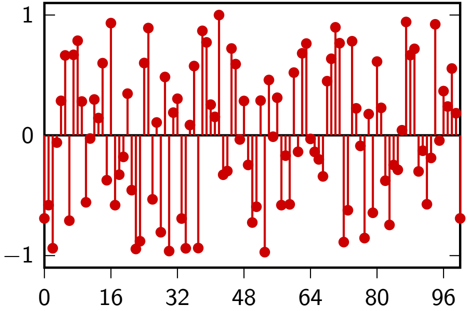
The Karplus-Strong Algorithm

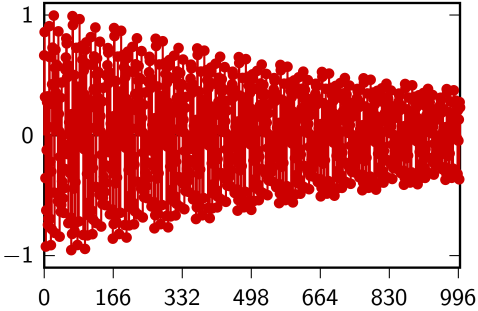
$M = 100$, $\alpha = 0.9$, $\bar{x}[n]$: 100 random values between 0 and 99, zero elsewhere



The Karplus-Strong Algorithm

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





Play

Recap

- 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!