

Memory Filter

Sinewave Generator

```
singen(M,N) :=
  for n ∈ 0 .. M - 1
    xn ← 100 · cos( n ·  $\frac{2 \cdot \pi}{N}$  )
  return x
```

N := 16 Samples per cycle

M := 30 · N Sample in signal

Generate input signal

Vin := singen(M,N) Sinewave signal

x0 := zero_{gen}($\frac{M}{4}$, N) Appended zero value vector

Vin := stack(Vin, x0) Input - Output Array

NN := rows(Vin) Size of arrays

n := 0 .. NN Sample array (for plotting)

Zero vector generator

```
zerogen(M,N) :=
  for n ∈ 0 .. M - 1
    xn ← 0
  return x
```

τ := 15 Filter time constant

Zn := (1 0 0 0 0 0 0 0 0) ^T Memory filter - Numerator

Zd := [K 0 0 0 0 0 0 0 (K - 1)] ^T Memory filter - denominator

RZd := polyroots(Zd)

$$RZd = \begin{pmatrix} -0.932 + 0.386i \\ -0.932 - 0.386i \\ -0.386 + 0.932i \\ -0.386 - 0.932i \\ 0.386 - 0.932i \\ 0.386 + 0.932i \\ 0.932 - 0.386i \\ 0.932 + 0.386i \end{pmatrix}$$

Roots on numerator all at zero

Roots of denominator are equally spaced around and inside the unit circle.

Digital Filtering Subroutine

"x" is an array of real numbers containing the input data.

Zn is a vector of coefficients for the filter numerator polynomial in ascending powers of 1/z.

Zd is a vector of coefficients for the filter denominator polynomial in ascending powers of 1/z.

Returned vector is the filtered data.

digitalfilter(x,Zn,Zd) :=

Nd ← rows(Zd)

Nn ← rows(Zn)

Nx ← rows(x)

for k ∈ 0..Nx

 yk ← 0

jn ← 0

jd ← 0

for n ∈ 0..Nx - 1

 z0 ← xn

 yn ← 0

 for k ∈ 0..jn

 yn ← yn + xn-k · Zn_k

 for k ∈ 1..jd

 yn ← yn - Zd_k · yn-k if jd > 0

 jn ← jn + 1 if jn < Nn - 1

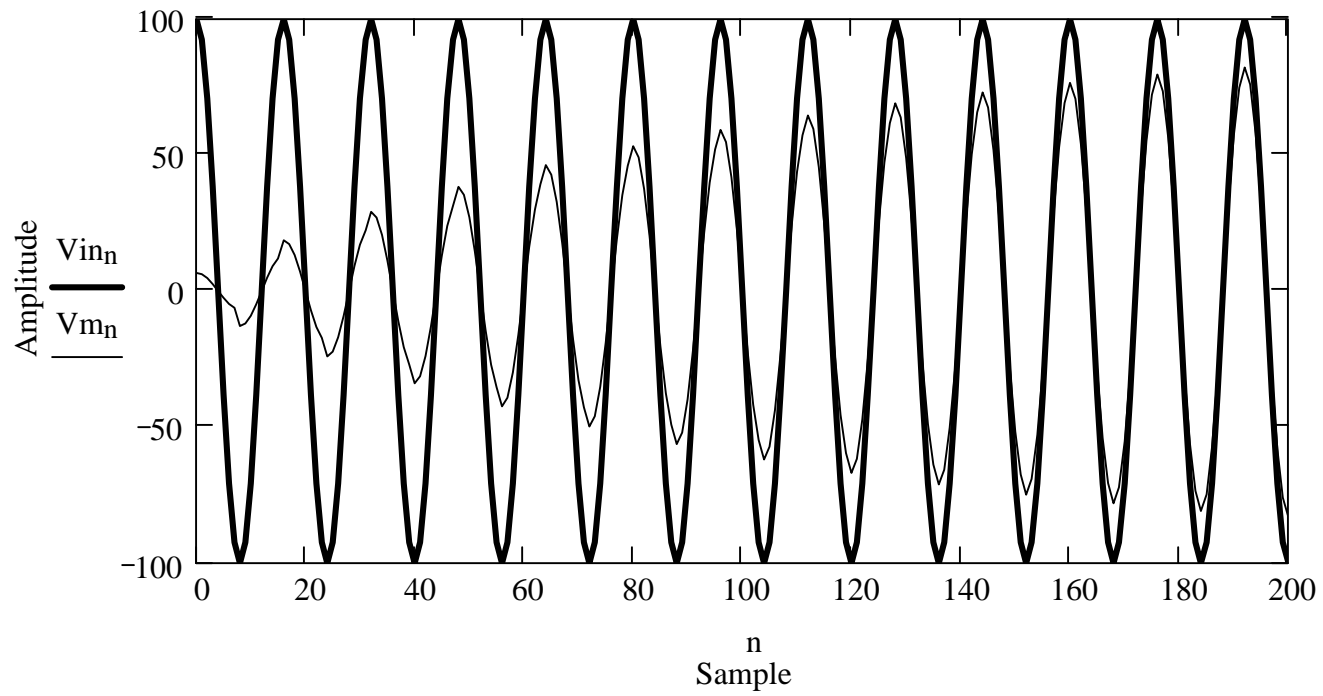
 jd ← jd + 1 if jd < Nd - 1

 yn ← $\frac{y_n}{Zd_0}$

```
n ← 0  
return y
```

$V_m := \text{digitalfilter}(V_{in}, Z_n, Z_d)$

Transient on startup



Transient on loss of signal

