CAPÍTULO 3.
Mis primeros filtros digitales

Sé que en este punto del libro ya querrás conocer ejemplos concretos de filtros digitales y su efecto en señales 1D (uni-dimensionales). Este Capítulo está diseñado precisamente para que empieces a filtrar señales 1D con filtros muy sencillos, conocidos como filtros de promedio. Adicionalmente, conocerás su contraparte IIR denominada Integrador Leaky.

Al finalizar el capítulo, deberás estar en capacidad de:

1. Diseñar filtros pasa-bajos para señales 1D, específicamente filtros de promedio.
2. Filtrar señales 1D con filtros de promedio.
3. Explicar el comportamiento en frecuencia de los filtros de promedio, tanto para valores de M par como impar.
4. Explicar las diferencias entre el filtro de promedio y el filtro Integrador Leaky.

3.1. INTRODUCCIÓN AL FILTRO DE PROMEDIO

Para entender en qué consiste el filtro de promedio, es necesario que previamente recordemos cómo se caracteriza un sistema Lineal e Invariante en el Tiempo (LTI). Específicamente, la salida del sistema, $y[n]$, se encuentra calculando la convolución entre la señal de entrada, $x[n]$, y la respuesta al impulso, $h[n]$.

Es decir, si el sistema es LTI, se cumple que:

$$y[n] = x[n] \otimes h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n - k] \qquad \text{Ecuación 12}$$

Adicionalmente, es necesario recordar en qué consiste convolucionar x[n] con un impulso ubicado en el origen, o desplazado, por ejemplo:

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

$$x[n] \otimes \delta[n - 1] = x[n - 1],$$

$$x[n] \otimes \delta[n - 2] = x[n - 2],$$

$$x[n] \otimes \delta[n + 1] = x[n + 1],$$

$$x[n] \otimes \delta[n + 2] = x[n + 2]$$

Entonces,

$$x[n] \otimes \delta[n - k] = x[n - k] \quad k \in Z \qquad \text{Ecuación 13}$$

Si unimos el concepto de la Ecuación 10 con el de la Ecuación 11, podremos identificar que si $y[n] = x[n - k]$, entonces su respuesta al impulso es $h[n] = \delta[n - k]$.

De forma general,

$$\text{si } y[n] = \sum_{k=0}^{M-1} x[n - k] \,,$$

$$\text{se tiene que } h[n] = \sum_{k=0}^{M-1} \delta[n - k] \,.$$

Ahora bien, si a la respuesta al impulso obtenida anteriormente la escalamos por el factor $1/M$, obtenemos un filtro de promedio causal.

> En resumen, un filtro de promedio (MAF: Moving Average Filter) es un sistema LTI cuya respuesta al impulso contiene M impulsos consecutivos de amplitud 1/M, que típicamente inicia en el origen y termina en M-1, donde M-1 corresponde al orden del filtro, y M es la cantidad de términos (pasados, presente y/o futuros) de la señal de entrada. El mínimo valor de M=2 (es decir, filtro de primer orden).

La respuesta al impulso de los filtros en promedio causales se define como:

$$h[n] = \frac{1}{M} \sum_{k=0}^{M-1} \delta[n - k] \qquad \text{Ecuación 14}$$

Cuya función de transferencia, es:

$$H(z) = \frac{1}{M} \sum_{k=0}^{M-1} z^{-k} \qquad \text{Ecuación 15}$$

Gráficamente, la respuesta al impulso de un filtro de promedio causal con M=11, es:

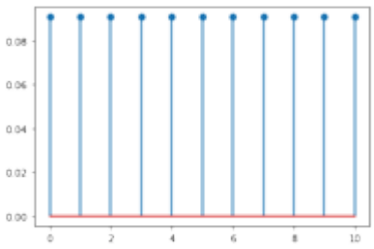

Figura 16. Respuesta al impulso de un filtro de promedio causal, M=11.

La cual se puede dibujar con el siguiente código en Python:

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
M=11
n = np.linspace(0,M-1,M)
x = np.ones([M])/M
plt.stem(n,x, use_line_collection="True")
```

Este filtro de promedio también puede ser simétrico respecto al origen. En ese caso, típicamente se trabaja con *M* impar, cuya respuesta al impulso se define así:

$$h[n] = \frac{1}{M}\sum_{k=-(M-1)/2}^{M-1} \delta[n - k] \qquad \text{Ecuación 16}$$

Suponiendo que M=11, entonces:

$$h[n] = \frac{1}{11}\sum_{k=-5}^{5} \delta[n - k]$$

El cual corresponde a un filtro de promedio no causal (Figura 17). Entonces, para calcular la salida del sistema es necesario conocer la entrada en el tiempo actual, cinco valores pasados y cinco valores futuros del tiempo actual. Es decir, $y[n] = 1/11\ \{x[n] + x[n - 1] + x[n - 2] + x[n - 3] + x[n - 4] + x[n - 5] + x[n + 1] + x[n + 2] + x[n + 3] + x[n + 4] + x[n + 5]\}$.

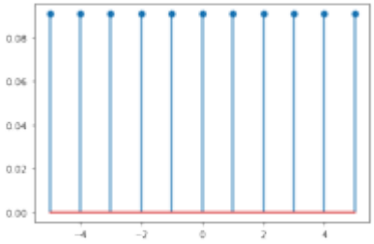

Figura 17. Respuesta al impulso de un filtro de promedio no causal, M=11.

Aunque el orden del filtro de la Figura 16 es el mismo del de la Figura 17, la principal diferencia radica en que en el primero se puede calcular la salida en tiempo real, mientras que, en el segundo es necesario que previamente se haya almacenado (o transmitido) la señal de entrada.

3.2. EFECTO DEL FILTRO DE PROMEDIO

El efecto del filtro de promedio en una señal 1D consiste en suavizarla, es decir, reducir los rizos que pueda contener la señal, manteniendo su *forma*. En otras palabras, el filtro de promedio actúa como un filtro pasa-bajos.

Para ilustrar este efecto, primero crearemos una señal senoidal a la cual le adicionaremos ruido, y posteriormente la filtraremos con filtros de promedio de diferente orden.

El código en Python paso a paso es el siguiente:

```
Paso 1:  importar librerías de trabajo

import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
from scipy import signal
import math

Paso 2:  generar una señal sin ruido

step = 0.001
t = np.arange(0,2,step)
fs = 1 / step
print(fs)
frecuencia = 2 # Hz
frad = frecuencia * 2 * math.pi
x1 = np.sin(frad*t)
plt.plot(t,x1)
plt.title('señal sin ruido')
```

La señal que se obtiene es una señal senoidal de 2 segundos de duración, con $f = 2\ Hz$, $f_s = 1\ kHz$, y amplitud en el rango [-1 1] (Ver Figura 18). Recordemos que utilizamos `plt.plot` para que tenga apariencia de señal continua, aunque realmente corresponde a una señal discreta.

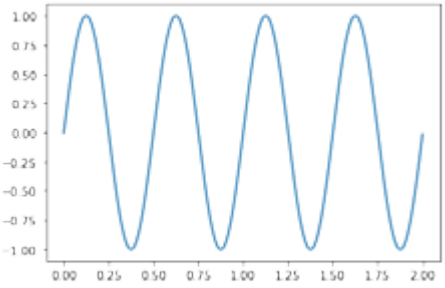

Figura 18. Señal senoidal sin ruido.

```
Paso 3:  generar ruido aleatorio

samples = len(x1)
An= 0.5
noise = An*np.random.rand(samples) - An/2
plt.plot(t,noise)
plt.title('Ruido')
```

En este paso se obtiene una señal que corresponde a ruido de 2 segundos de duración, cuya amplitud se encuentra en el rango [-0.25 0.25)]. (Ver Figura 19).

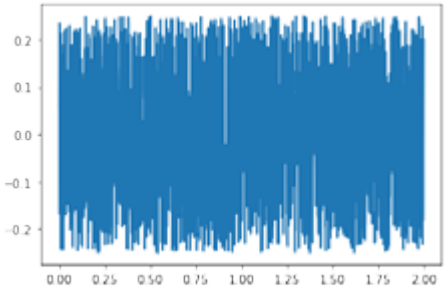

Figura 19. Ruido aleatorio.

```
Paso 4:  sumar la señal senoidal con la señal de ruido

xnoise = x1 +  noise
plt.plot(t,xnoise)
plt.title('Señal con ruido')
```

La nueva señal (Figura 20) corresponde a una señal senoidal con ruido de fondo, conservando la frecuencia fundamental de la señal de la Figura 18. No obstante, la amplitud está ahora en el rango [-1.25 1.25)].

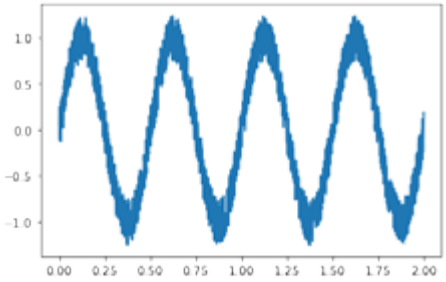

Figura 20. Señal senoidal con ruido de fondo.

```
Paso 5:  aplicación de filtro de promedio (M=7, 11, y 111)

a = 1
M = 7
b7 = np.ones([M])/M
y7 = signal.filtfilt(b7, a, xnoise)

M = 11
b11 = np.ones([M])/M
y11 = signal.filtfilt(b11, a, xnoise)

M = 111
b111 = np.ones([M])/M
y111 = signal.filtfilt(b111, a, xnoise)

# Se grafican los resultados
plt.rcParams["figure.figsize"] = (20,10)
plt.subplot(2,2,1)
plt.plot(t,xnoise)
plt.title('a)')
plt.subplot(2,2,2)
plt.plot(t,y7)
```

```
plt.title('b)')
plt.subplot(2,2,3)
plt.plot(t,y11)
plt.title('c)')
plt.subplot(2,2,4)
plt.plot(t,y111)
plt.title('d)')
plt.show()
```

Se utiliza la instrucción *filtfilt* de la librería signal para aplicar el filtro previamente diseñado (con `np.ones([M])/M`) a la señal *xnoise*. Esta instrucción permite filtrar señales 1D con filtros FIR o IRR. En el caso de filtros FIR, como corresponde al filtro de promedio, es necesario trabajar con `a = 1`.

Como resultado del código anterior se obtienen cuatro sub-gráficas, las cuales se presentan en la Figura 21.

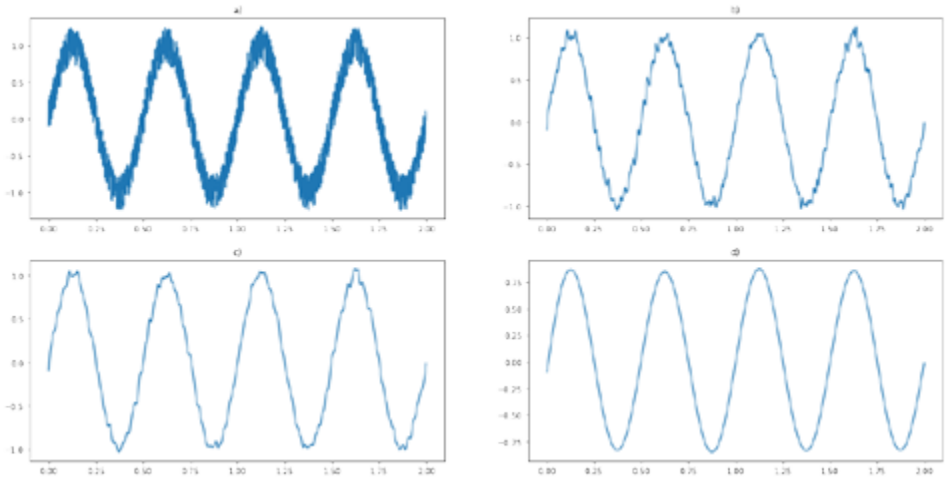

Figura 21. Resultado de filtrar una señal senoidal ruidosa con un filtro de promedio: a) señal de entrada, b) señal filtrada con M=7, c) señal filtrada con M=11, d) señal filtrada con M=111.

Al comparar los resultados obtenidos con filtros de promedio con diferentes M, se aprecia que a medida que M aumenta el efecto de suavizado es mayor, es decir, se reduce en mayor medida el rizado (ruido de fondo) de la señal. No obstante, como veremos en la siguiente sección, no se recomienda aumentar abruptamente el orden del filtro, porque se puede producir un efecto no deseado al eliminar componentes de frecuencia de la señal que son importantes. Se sugiere que el estudiante utilice un M alto (por ejemplo, M=501), y obtenga sus propias conclusiones del efecto del filtro sobre la señal.

3.3. RESPUESTA EN FRECUENCIA DEL FILTRO DE PROMEDIO

En esta sección nos centraremos en conocer y comprender el impacto que tiene el valor de M en la respuesta en frecuencia del filtro de promedio. El filtro MAF es

un filtro pasa-bajos cuya frecuencia de corte disminuye a medida que aumenta el valor de M. A diferencia de los filtros análogos, en los que la frecuencia de corte la expresamos (típicamente) en Hz, en el caso de los filtros digitales, esta frecuencia se encuentra normalizada en el rango $[0 \quad \pi)]$, por ejemplo 0.2π, con unidades $[rad/muestra]$. El rango total de la respuesta en frecuencia del filtro digital (bilateral) corresponde a $[-\pi \quad \pi)]$.

Como primer paso, vamos a reescribir la respuesta al impulso del filtro de promedio, de la forma:

$$\frac{h[n] = u[n] - u[n-M]}{M} \qquad \text{Ecuación 17}$$

donde M - 1 es el orden del filtro. Este resultado es equivalente al obtenido en la Ecuación 14.

Como segundo paso, vamos a calcular la DTFT (Transformada de Fourier de Tiempo Discreto) de $h[n]$, es decir:

$$h[n] \xrightarrow{DTFT} H(e^{j\omega}) \qquad \text{Ecuación 18}$$

Obteniendo que,

$$|H(e^{j\omega})| = \frac{1}{M}\left|\frac{sin\left(\frac{M*\omega}{2}\right)}{sin\left(\frac{\omega}{2}\right)}\right| \qquad \text{Ecuación 19}$$

Cuando graficamos la magnitud de la respuesta en frecuencia del filtro de promedio, encontramos que presenta un comportamiento especial, que lo podemos resumir como:

- Todos los filtros de promedio tienen un lóbulo principal alrededor de $\omega = 0$, y varios lóbulos secundarios que inician en $-\pi$ y terminan en π.
- La amplitud de los lóbulos secundarios disminuye a medida que se alejan de $\omega=0$. Cada lóbulo secundario es más pequeño que su antecesor (entre $[0$ $\pi]$) y existe un efecto espejo con las frecuencias negativas.
- La cantidad de lóbulos en el rango $[-\pi \quad \pi]$ es igual a M-1. Hay un lóbulo principal y M-2 lóbulos secundarios.
- Si el filtro tiene un M par, el primer "cruce por cero" y el último "cruce por cero" ocurren en las frecuencias $-\pi$ y π, respectivamente. En caso contrario, si M es impar, en esas frecuencias no existirá cruce por cero.
- En todos los casos, los cruces por cero se encuentran ubicados en $2\pi k/M$. El rango de k es $[1 \quad (M-1)/2]$ para M impar, y $[1 \quad M/2]$ para M par.

Nota: se dibuja la magnitud de la respuesta en frecuencia del filtro, por lo cual no tendrá valores negativos y formalmente no existirán los "cruces por cero". Sin embargo, sí existen valores en los cuales la amplitud ha disminuido, llega a cero, y vuelve a aumentar, los consideraremos como "cruces por cero".

Por ejemplo, para $M = 7$, el filtro de promedio tiene un lóbulo principal y cinco lóbulos secundarios, de los cuales dos lóbulos y medio (secundarios) se encuentran en las frecuencias positivas (Ver Figura 22). Dado que M es impar, el último cruce por cero en las frecuencias positivas (al igual que el primer cruce por cero en las frecuencias negativas) no ocurre en $\omega=\pi$.

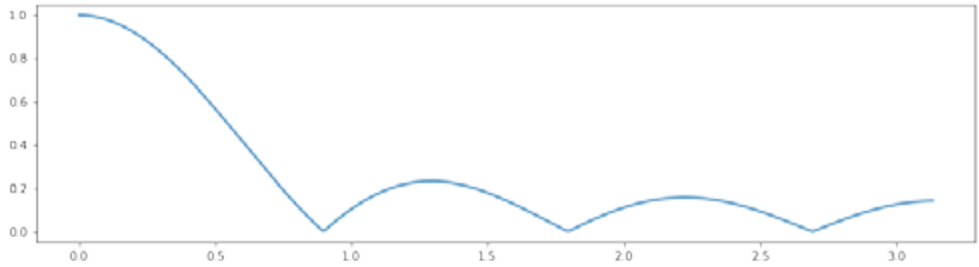

Figura 22. Magnitud de la respuesta en frecuencia de un filtro de promedio, M=7.

Para obtener la gráfica de la magnitud de la respuesta en frecuencia del filtro MAF, utilizamos el siguiente código en Python:

```
from scipy import signal
import numpy as np
from scipy import signal
import matplotlib.pyplot as plt
import math
M = 7
M7 = np.ones([M])/M
a = 1
w1, v1 = signal.freqz(M7, a)
plt.rcParams["figure.figsize"] = (14,8)
ax = plt.subplot(2, 1, 1)
plt.plot(w1, np.abs(v1))
plt.title('Respuesta en frecuencia filtro digital de promedio, M=7')
```

La instrucción `signal.freqz` de la librería de *scipy* de Python permite graficar la respuesta en frecuencia de filtros digitales, tanto FIR como IIR. Las entradas de esta instrucción corresponden a los coeficientes de los polinomios tanto del numerador como del denominador de la función de transferencia del filtro digital. En el caso del filtro MAF, por ser un filtro FIR, el denominador es una constante igual a uno, y entonces, a la entrada "`a`" de la instrucción `signal.freqz` le asignamos el valor de uno. El resultado corresponde al vector de frecuencias normalizadas, `w1`, y al vector de amplitudes, `v1`. Con la instrucción `np.abs(v1)` se calcula la magnitud de la respuesta en frecuencia del filtro digital.

Podemos obtener cada uno de los cruces por cero del filtro de promedio, con el siguiente código en lenguaje Python:

```
M=7

# k=1, entonces
k=1
wc1= 2*math.pi/M
print("frecuencia cruce por cero 1:", wc1)

# k=2, entonces
k=2
wc2= 2*math.pi*k/M
print("frecuencia cruce por cero 2:", wc2)

# k=3, entonces
k=3
wc3= 2*math.pi*k/M
print("frecuencia cruce por cero 3:", wc3)
```

y obtendríamos:

```
frecuencia cruce por cero 1: 0.8975979010256552
frecuencia cruce por cero 2: 1.7951958020513104
frecuencia cruce por cero 3: 2.6927937030769655
```

Cuando el valor de *M* es alto, se recomienda utilizar una estructura anidada (por ejemplo, ciclo for) para encontrar los cruces por cero del filtro digital, así:

```
M = 7
for k in range(1,int((M-1)/2)+1):
    wc= 2*3.14*k/M
    print("frecuencia de cruce por cero",k, ":", wc)
```

Supongamos ahora que nuestro filtro trabaja con $M = 31$, ¿cuántos lóbulos secundarios tendrá? La respuesta es 29 lóbulos secundarios, por lo que, de forma similar al caso anterior no se encontrarán cruces por cero en $-\pi$, ni en π. La gráfica se presenta en la Figura 23.

Figura 23. Magnitud de la respuesta en frecuencia de un filtro de promedio, M=31.

Independiente del orden del filtro, tendremos que en $\omega = 0$ la amplitud es igual a uno. Se aprecia que de forma similar a la gráfica de la Figura 22, el último lóbulo queda a "la mitad", es decir, no llega a cero.

Los cruces por cero los obtenemos con el siguiente código en Python:

```
M = 31
for k in range(1,int((M-1)/2)+1):
    wc= 2*3.14*k/M
    print("frecuencia de cruce por cero",k, ":", wc)
```

```
frecuencia de cruce por cero 1: 0.20258064516129032
frecuencia de cruce por cero 2: 0.40516129032258064
frecuencia de cruce por cero 3: 0.607741935483871
frecuencia de cruce por cero 4: 0.8103225806451613
frecuencia de cruce por cero 5: 1.0129032258064516
frecuencia de cruce por cero 6: 1.215483870967742
frecuencia de cruce por cero 7: 1.4180645161290324
frecuencia de cruce por cero 8: 1.6206451612903225
frecuencia de cruce por cero 9: 1.823225806451613
frecuencia de cruce por cero 10: 2.0258064516129033
frecuencia de cruce por cero 11: 2.2283870967741937
frecuencia de cruce por cero 12: 2.430967741935484
frecuencia de cruce por cero 13: 2.6335483870967744
frecuencia de cruce por cero 14: 2.836129032258065
frecuencia de cruce por cero 15: 3.0387096774193547
```

Como tercer ejemplo, utilizaremos un filtro con M par. Específicamente, si $M = 8$, obtendremos una gráfica que contiene tres lóbulos secundarios en las frecuencias positivas, y el último cruce por cero ocurre exactamente en $\omega = \pi$ (Ver Figura 24).

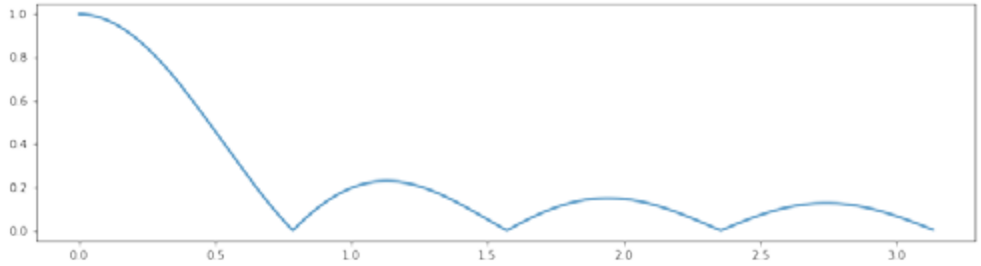

Figura 24. Magnitud de la respuesta en frecuencia de un filtro de promedio, M=8.

La respuesta en frecuencia se obtiene con el siguiente código en Python:

```
from scipy import signal
M = 8
M8 = np.ones([M])/M
a = 1
w1, v1 = signal.freqz(M8, a)
plt.rcParams["figure.figsize"] = (14,8)
ax = plt.subplot(2, 1, 1)
plt.plot(w1, np.abs(v1))
```

Y los cruces por cero, así:

```
M = 8
for k in range(1,int((M)/2)+1):
    wc= 2*3.14*k/M
    print("frecuencia de cruce por cero",k,":", wc)
```

```
frecuencia de cruce por cero 1: 0.785
frecuencia de cruce por cero 2: 1.57
frecuencia de cruce por cero 3: 2.355
frecuencia de cruce por cero 4: 3.14
```

Finalmente, si comparamos la primera frecuencia de cruce por cero de los filtros de promedio con $M = 7$, $M = 8$, y $M = 31$, podemos concluir que a medida que M aumenta, la frecuencia del primer cruce por cero disminuye (es decir, cuando utilizamos un orden de filtro alto, la frecuencia de corte es baja). No obstante, independiente del valor de M, el filtro de promedio se comporta como un filtro pasa-bajos.

3.4. FILTRO INTEGRADOR LEAKY

Este filtro tiene un comportamiento parecido al filtro de promedio (efecto pasa-bajo) cuando $M \geq 100$. La ecuación de entrada-salida se define como:

$$y[n] = \lambda y[n - 1] + (1 - \lambda)x[n] \qquad \text{Ecuación 20}$$

Cuya relación de λ y M está dada por:

$$\lambda = \frac{M - 1}{M} \qquad \text{Ecuación 21}$$

De tal forma que si $M = 100$, entonces $\lambda = 99/100 = 0.99$.

Por lo que, para este caso específico la salida es:

$$y[n] = 0.99y[n - 1] + 0.01x[n]$$

Esto significa que, para obtener la salida en el momento actual se conserva en gran parte la salida del momento anterior, y solamente una pequeñísima parte de la entrada en el momento actual. Adicionalmente, si la entrada solo existe en un momento específico (ej. $x[n] = \delta[n]$), la salida será distinta de cero a partir de ese momento en adelante.

Veamos precisamente cuál es la respuesta al impulso del filtro Leaky.

Reescribamos la ecuación 18 de la siguiente forma:

$$y[n] = \lambda y[n - 1] + (1-\lambda)\delta[n] \qquad \text{Ecuación 22}$$

Y supongamos que el sistema inicia en $n = 0$, es decir que antes de ese momento tanto la entrada como la salida eran de amplitud igual a cero.

Entonces,

- $y(0) = 0.99y(-1) + 0.01\delta[n]$, que es equivalente a $y(0) = 0.01$, dado que $y(-1) = 0$, y $x(0) = 0.01$.
- $y(1) = 0.99y(0)$, que es equivalente a $y(1) = 0.99 * 0.01$, dado que $x(1) = 0$.
- $y(2) = 0.99y(1)$, que es equivalente a $y(2) = 0.99 * 0.99 * 0.01$, dado que $x(2) = 0$.
- $y(3) = 0.99y(2)$, que es equivalente a $y(3) = 0.99 * 0.99 * 0.99 * 0.01$, dado que $x(3) = 0$.
- .... $y(k)=0.99y(k-1)$, que es equivalente a $y(k) = 0.99^k * 0.01$.

De forma general, la respuesta al impulso del Filtro Leaky se expresa como:

$$h[n] = \lambda^n (1 - \lambda) \qquad \text{para } n \geq 0 \qquad \text{Ecuación 23}$$

Examinando la ecuación 23 podemos concluir que este filtro es de respuesta al impulso infinita, dado que, a partir de $n = 0$, las amplitudes de $h[n]$ serán distintas de cero.

A continuación, dibujaremos la ecuación de entrada-salida, utilizando un diagrama de bloques:

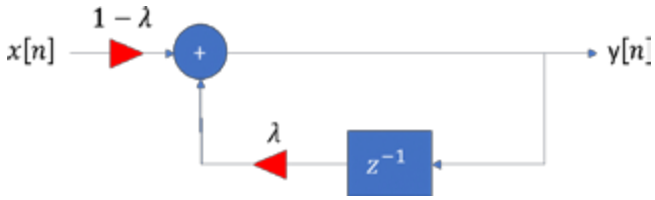

Figura 25. Diagrama de bloques filtro Leaky.

Para obtener la salida con este tipo de filtros, se necesita una unidad de retardo, un sumador y dos multiplicadores, independiente de M.

Ahora bien, dibujemos el diagrama de bloques para un filtro de promedio con $M = 100$, y comparemos el uso de recursos.

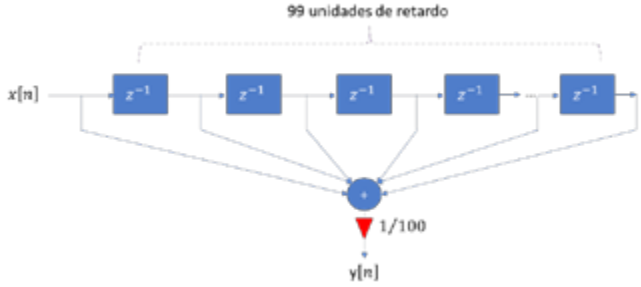

Figura 26. Diagrama de bloques filtro de promedio, M=100.

En este caso, se necesitan 99 unidades de retardo, un sumador y un multiplicador. Es claro que la cantidad de unidades de retardo es significativamente mayor que en el filtro Leaky.

Lo que significa que si se quiere implementar un filtro que *promedie* el comportamiento de la señal de entrada en las últimas 100 o 1000 muestras (por ejemplo), es más eficiente a nivel computacional utilizar una estructura como la de la Figura 25, que como de la Figura 26.

3.5. GENERALIDADES DE LOS FILTROS DIGITALES

Iniciaremos este subcapítulo de generalidades de los filtros digitales, clasificándolos en relación con su respuesta al impulso. Primero en términos de duración, y segundo en términos de cómputo. Posteriormente, revisaremos la definición de estabilidad de los filtros digitales, tomando como ejemplo el filtro de promedio y el *Leaky*.

Clasificación de los filtros digitales:

Si la respuesta al impulso del filtro es de duración finita, decimos que es FIR. En caso contrario, decimos que el filtro es IIR.

Por otro lado, un filtro puede ser causal o no causal. Un filtro es causal si su salida depende de la entrada en el mismo valor de tiempo (discreto) y/o de valores pasados de tiempo.Y es no causal, si la salida depende de valores futuros de la señal de entrada. Un filtro causal se puede ejecutar en tiempo real, es decir, que a medida que ingresa la entrada al sistema se calcula su salida. Mientras que, en filtros no causales, necesitamos conocer toda la señal de entrada para calcular la salida del sistema.

Combinando las dos clasificaciones anteriores, se pueden tener filtros FIR causales, FIR no causales, IIR causales e IIR no causales. Puedes revisar ejemplos de cada caso en el Capítulo 2.3.

Estabilidad de los filtros digitales:

Un filtro es estable si la salida del filtro es acotada para entradas acotadas. Es decir, si se cumple con la siguiente condición:

Sea $|x[n]| < M$, $|y[n]| < P$, para $M,P < \infty$. Entonces $\sum n|h[n]| < L$ para $L<\infty$.

De tal forma que,TODOS los filtros FIR son estables. Por lo que, todos los filtros de promedio son estables.

Vamos ahora a revisar la estabilidad en los filtros Leaky. Recordemos que su res-

puesta al impulso es de la forma $h[n] = \lambda^n (1 - \lambda)$ para $n \geq 0$. Entonces es necesario evaluar dos posibles escenarios, cuando $|\lambda| < 1$ y cuando $|\lambda| \geq 1$.

Escenario 1: $|\lambda| < 1$

En este caso, $\sum_{n=-\infty}^{\infty}|h[n]| = |1 - \lambda| \sum_{n=-\infty}^{\infty} |\lambda^n| = |1 - \lambda|\{\lambda^0+\lambda^1+\lambda^2+\lambda^3+\cdots+\lambda^\infty\}$ es un valor finito, dado que cada vez se suma un término más pequeño que el anterior. Por ejemplo, supongamos que $\lambda = 0.5$, entonces $|1 - \lambda| \sum_{n=-\infty}^{\infty}|\lambda^n| = |0.5|\{0.5^0+ 0.5^1 + 0.5^2 + 0.5^3+\cdots+0.5^\infty\} = \frac{0.5 + 0.25 + 0.125 + 0.0625+\cdots}{2} = \frac{1}{2}$ Entonces, el filtro IIR es estable.

Escenario 2: $|\lambda| \geq 1$

En este caso, $\sum_{n=-\infty}^{\infty}|h[n]| = |1 - \lambda| \sum_{n=-\infty}^{\infty} |\lambda^n| = |1 - \lambda|\{\lambda^0+\lambda^1+\lambda^2+\lambda^3+\cdots+\lambda^\infty\}$ es un valor infinito, dado que cada vez se suma un término más grande que el anterior. Por ejemplo, supongamos que $\lambda = 2$, entonces $|1 - \lambda| \sum_{n=-\infty}^{\infty}|\lambda^n| = |-1|\{2^0+ 2^1 + 2^2 + 2^3+\cdots+2^\infty\} =1 + 2 +4 + 8 + 16 + \cdots + \rightarrow\infty$. Entonces, el filtro IIR es inestable.

En resumen, algunos filtros IIR son estables, y otros son inestables. En el caso del filtro Leaky, es estable siempre y cuando se cumpla que $|\lambda| < 1$.