Sistemas discretos y ecuaciones en diferencias

Una ecuación en diferencias describe la relación entre la señal de salida $y[n]$ y la señal de entrada $x[n]$. incluyendo valores pasados de ambas. Este tipo de representación permite caracterizar filtros digitales, estudiar su respuesta a distintas señales de prueba y analizar propiedades como causalidad, estabilidad, energía y potencia de la salida.

La forma general de una ecuación en diferencias lineal e invariante en el tiempo (LTI) es:

$$y[n] = \sum_{k=0}^{M} b_k \, x[n-k] + \sum_{k=1}^{N} a_k \, y[n-k]$$

Donde:

- $x[n]$: entrada del sistema
- $y[n]$: salida del sistema
- b_k: coeficientes asociados a la parte no recursiva
- a_k: coeficientes asociados a la carte recursiva

Según la estructura de la ecuación, los sistemas se clasifican en dos tipos:

FIR (Finite Impulse Response)

- La respuesta al impulso dura un número finito de muestras.
- La salida depende únicamente de entradas pasadas y presentes.
- Se llaman filtros no recursivos.

Por ejemplo:

$$y[n] = 0.5x[n] + 0.5x[n-1]$$

IIR (Infinite Impulse Response)

- La respuesta al impulso dura un número infinito de muestras.
- La salida depende tanto de entradas como de salidas pasadas.
- Se llaman filtros recursivos.

Por ejemplo:

$$y[n] = 0.5x[n] + 0.3y[n-1]$$

Respuesta al Impulso

La respuesta al impulso se define como la salida del sistema cuando la entrada es el delta unitario:

$$\delta[n] = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases}$$

Y en sistemas LTI se cumple que:

$$y[n] = (x * h)[n] = \sum_{k=-\infty}^{\infty} x[k]\, h[n-k]$$

Esto significa que una vez conocida $h[n]$, la salida ante cualquier entrada $x[n]$ se obtiene mediante convolución discreta.

Ejercicio 1:

Dada la siguiente ecuación en diferencias que modela un sistema LTI:

$$y[n] = 3 \cdot 10^{-2} \cdot x[n] + 5 \cdot 10^{-2} \cdot x[n-1] + 3 \cdot 10^{-2} \cdot x[n-2] + 1.5 \cdot y[n-1] - 0.5 \cdot y$$

- Graficar la señal de salida para cada una de las señales de entrada que generó en el TS1. Considere que las mismas son causales.
- Hallar la respuesta al impulso y usando la misma, repetir la generación de la señal de salida para alguna de las señales de entrada consideradas en el punto anterior.
- En cada caso indique la frecuencia de muestreo, el tiempo de simulación y la potencia o energía de la señal de salida.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sig
plt.close('all')

#---------------Funciones auxiliares---------------#
def sen(vmax, dc, ff, ph, N, fs):
    """Genera una senoidal de amplitud vmax, offset dc, freq ff (Hz), fase ph (r
    Ts = 1/fs
    t = np.linspace(0, (N-1)*Ts, N)
    x = vmax*np.sin(2*np.pi*ff*t + ph) + dc
    return t, x

def potencia(x):
    return np.mean(np.abs(x)**2)

def energia(x):
    return np.sum(np.abs(x)**2)

#---------------Sistema---------------#
b = np.array([0.03, 0.05, 0.03]) #para lo que tiene x
a = np.array([1, -1.5, 0.5]) #para lo que tiene y

N = 1000
fs = 80000
```

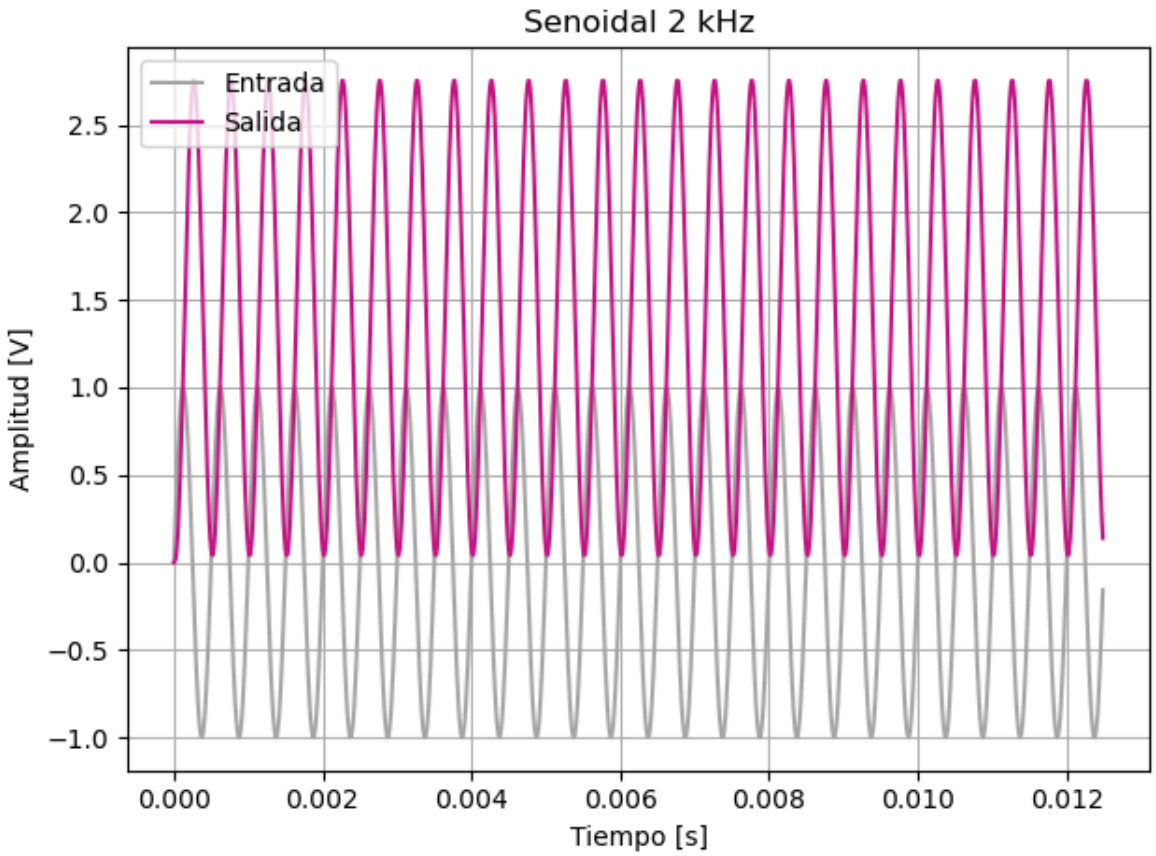
```
Ts = 1/fs
Tsim = N/fs

print("Frecuencia de muestreo fs =", fs, "Hz")
print("Cantidad de muestras N =", N)
print("Tiempo de simulación T =", Tsim, "s")

#---------------mis señales del TS1--------------#
t, x  = sen(1, 0, 2000, 0, N, fs)
t1, x1 = sen(2, 0, 2000, np.pi/2, N, fs)
tm, xm = sen(1, 0, 1000, 0, N, fs)
modulada = x*xm
P = potencia(x)
amplitudRecortada = np.sqrt(2*0.75*P)
recortada = np.clip(x, -amplitudRecortada, amplitudRecortada)
tq = np.linspace(0, (N-1)/fs, N)
cuadrada = sig.square(2*np.pi*4000*tq)
TsP = 0.01   # 10 ms
fsP = 80000
tp = np.linspace(0, Tsim, N, endpoint=False)
pulso = np.where((t >= 0) & (t <= TsP), 1, 0)

#---------------Entradas---------------#
entradas = [ #estpy literalmente armando un struct de c en python
    ("Senoidal 2 kHz", t, x),
    ("Senoidal amplificada y desfasada", t1, x1),
    ("Modulada", t, modulada),
    ("Recortada", t, recortada),
    ("Cuadrada 4 kHz", tq, cuadrada),
    ("Pulso rectangular 10 ms", tp, pulso),
]

#---------------Paso mis señales por la ecu de dif---------------#
salidas = [] #es la lista vacia
for nombre, tt, xx in entradas:
    y = sig.lfilter(b, a, xx) #me aplica la ecu en diferencias
    salidas.append((nombre, tt, xx, y)) #append me mete cosas en la lista
    print("#--- " + nombre + " ---#")
    print("\nPotencia salida =", potencia(y))
    print("\nEnergía salida =", energia(y))
    print("\n")
```

```
Frecuencia de muestreo fs = 80000 Hz
Cantidad de muestras N = 1000
Tiempo de simulación T = 0.0125 s
#--- Senoidal 2 kHz ---#

Potencia salida = 2.8776432773855847

Energía salida = 2877.6432773855845


#--- Senoidal amplificada y desfasada ---#

Potencia salida = 3.745276533393584

Energía salida = 3745.276533393584


#--- Modulada ---#

Potencia salida = 1.0617208497301638

Energía salida = 1061.7208497301638


#--- Recortada ---#

Potencia salida = 2.594253898020217

Energía salida = 2594.253898020217


#--- Cuadrada 4 kHz ---#

Potencia salida = 23.77707327551082

Energía salida = 23777.07327551082


#--- Pulso rectangular 10 ms ---#

Potencia salida = 14424.253306666018

Energía salida = 14424253.306666018
```

Senoidal 2 kHz

In [2]:
```
plt.figure(1)
plt.plot(t, x, label="Entrada", color="darkgray")
plt.plot(t, salidas[0][3], label="Salida", color="mediumvioletred")
#con salida[x][y] lo que hago es tomar el elemento (que es un struct) nro x de l
# y luego dentro de ese struct tomo el elemento nro y

plt.title("Senoidal 2 kHz")
plt.xlabel("Tiempo [s]")
plt.ylabel("Amplitud [V]")
plt.legend()
plt.grid(True)
```

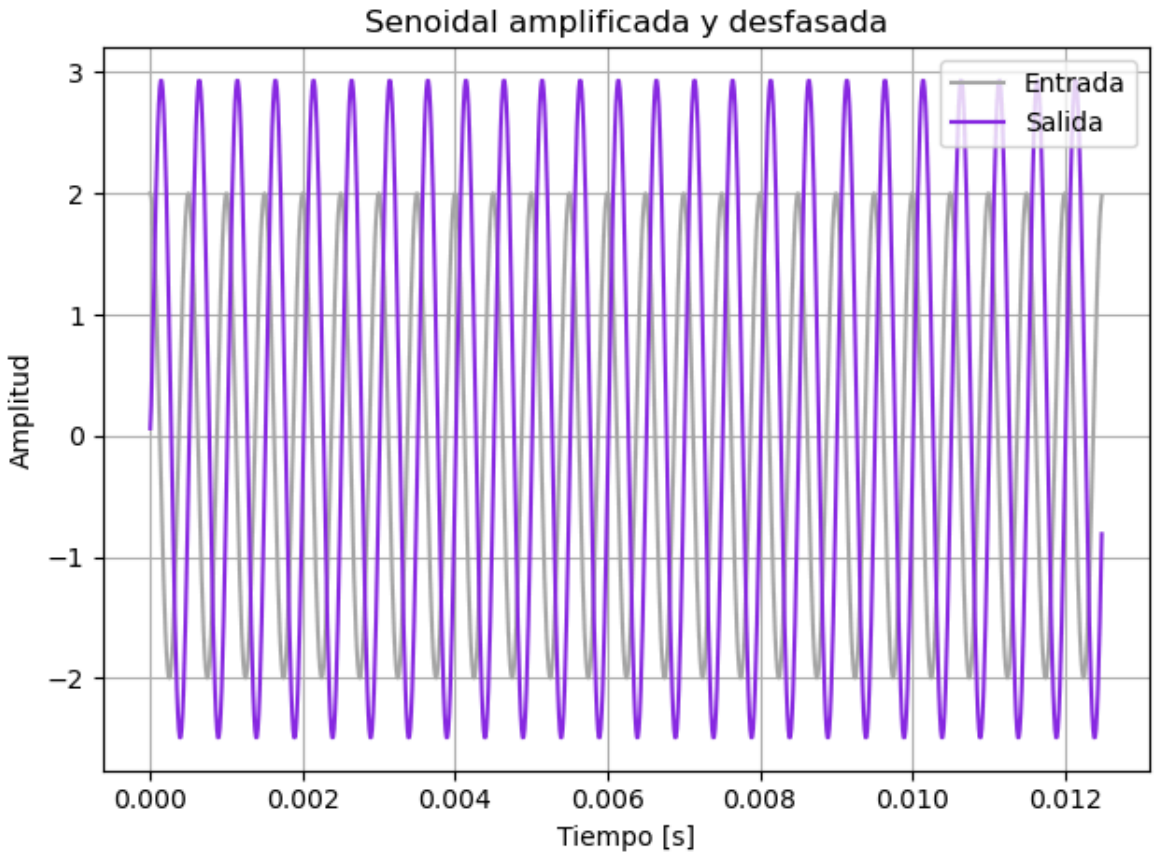
```
plt.tight_layout()
plt.show()
```



- En la salida se observa que la amplitud crece respecto de la entrada (osea como si el sistema amplificara esa frecuencia)
- El sistema no introduce distorsión armónica en señales senoidales, solo modifica ganancia y fase
- El sistema se comporta como un filtro que resuena en la zona de 2 kHz, amplificando esa componente.

Senoidal amplificada y desfasada

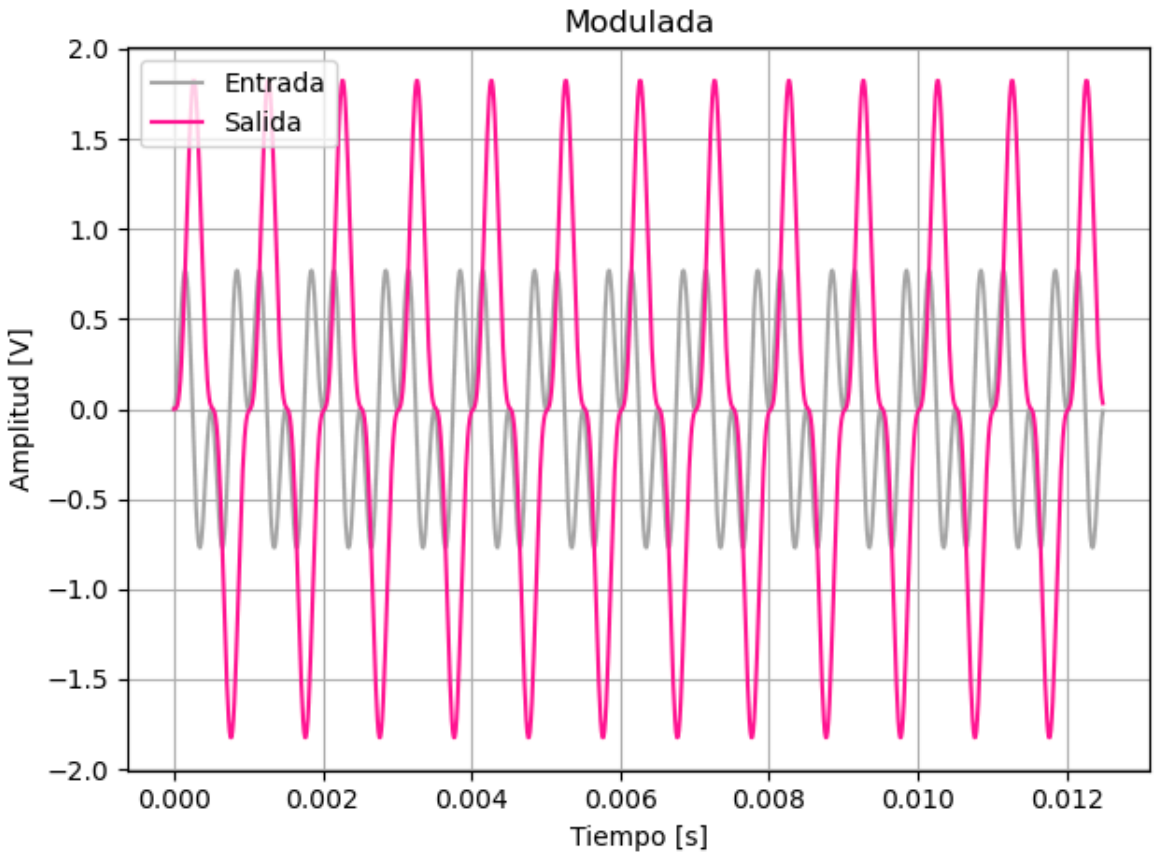
In [3]:
```
plt.figure(2)
plt.plot(t1, x1, label="Entrada", color="darkgray")
plt.plot(t1, salidas[1][3], label="Salida", color="blueviolet")

plt.title("Senoidal amplificada y desfasada")
plt.xlabel("Tiempo [s]")
plt.ylabel("Amplitud [V]")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



- La salida hace lo mismo que en el caso anterior, amplificaca y un corre la fase
- Amplifica y desplaza en fase pero no cambia la forma de la sinusoide, es un sistema lineal confirmado

Señal modulada

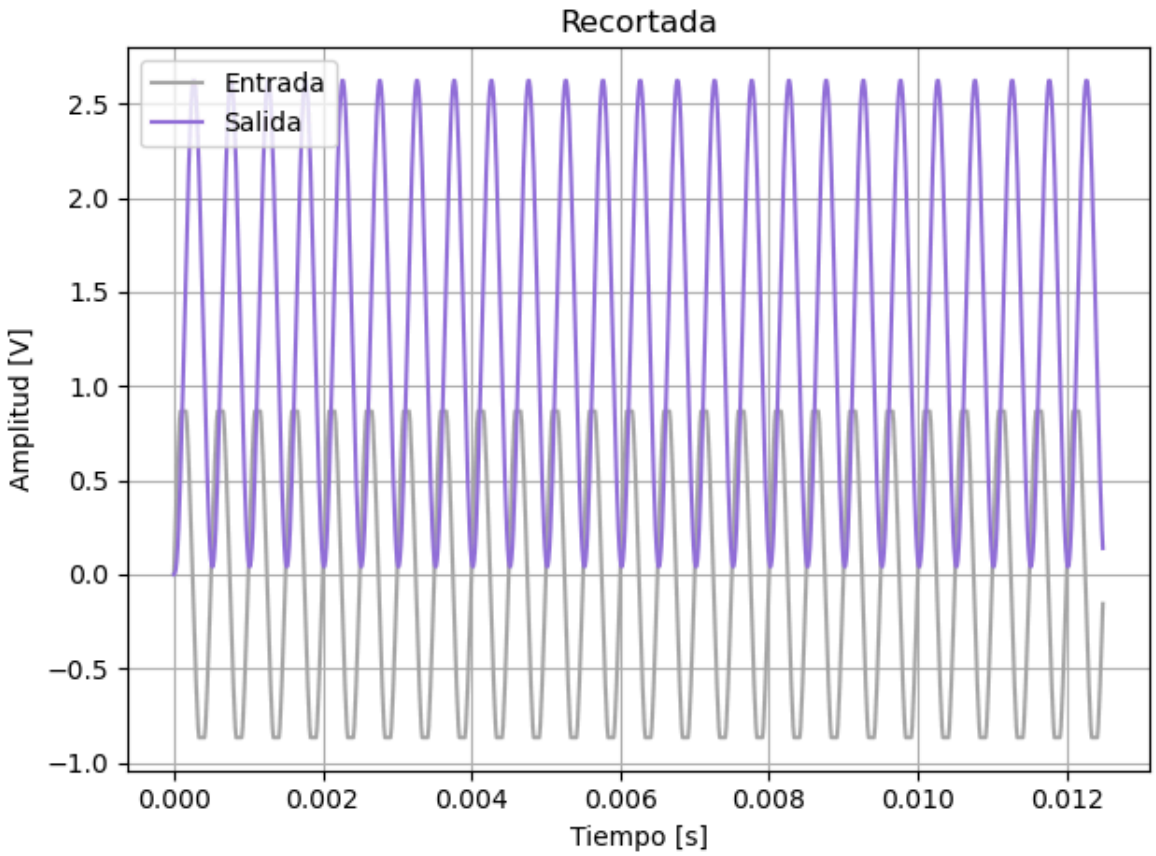
In [4]:
```
plt.figure(3)
plt.plot(t, modulada, label="Entrada", color="darkgray")
plt.plot(t, salidas[2][3], label="Salida", color="deeppink")

plt.title("Modulada")
plt.xlabel("Tiempo [s]")
plt.ylabel("Amplitud [V]")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



- La salida muestra que el sistema no responde igual a todas las frecuencias porque algunas se atenúan y otras se realzan
- el sistema actúa como filtro selectivo y deforma la modulación

Señal recortada

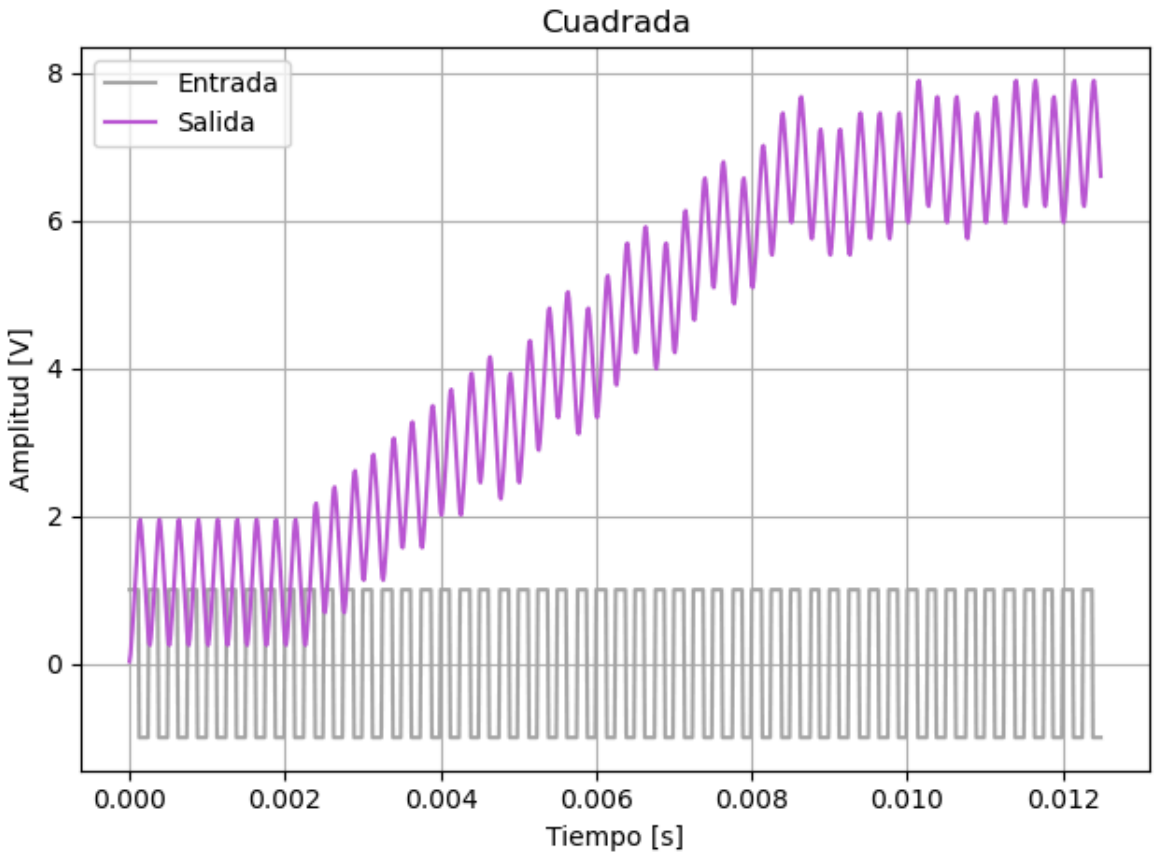
In [5]:
```
plt.figure(4)
plt.plot(t, recortada, label="Entrada", color="darkgray")
plt.plot(t, salidas[3][3], label="Salida", color="mediumpurple")

plt.title("Recortada")
plt.xlabel("Tiempo [s]")
plt.ylabel("Amplitud [V]")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



- En la salida desaparecen varios armónicos y queda algo parecido a una senoidal pura amplificada
- el sistema actúa como un filtro suavizante, eliminando lo que parecia una palnicie por el recorte

Señal cuadrada de 4 kHz

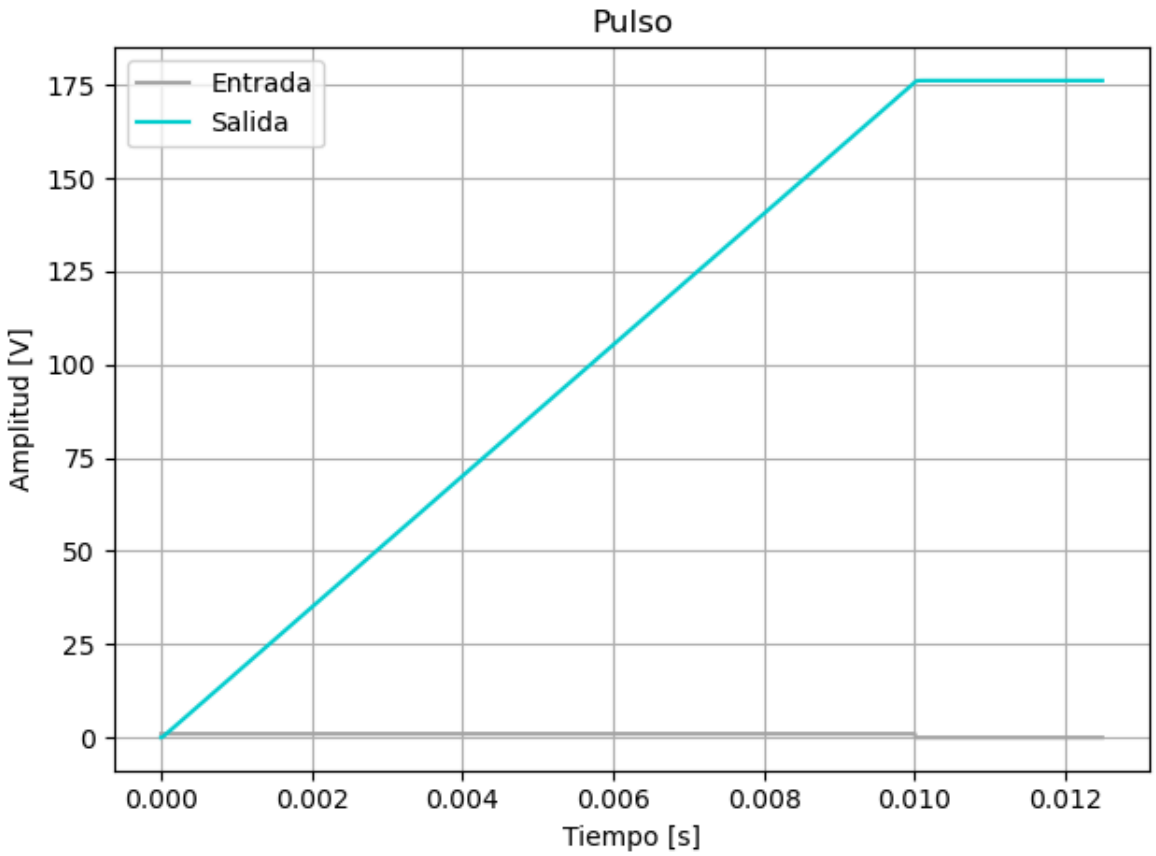
In [6]:
```
plt.figure(5)
plt.plot(tq, cuadrada, label="Entrada", color="darkgray")
plt.plot(tq, salidas[4][3], label="Salida", color="mediumorchid")

plt.title("Cuadrada")
plt.xlabel("Tiempo [s]")
plt.ylabel("Amplitud [V]")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



La función usada (lfilter) arranca suponiendo que las condiciones iniciales de la memoria son cero. Eso a veces genera un transitorio largo donde la salida parece crecer en una dirección hasta que el sistema se acomoda bien. Si el sistema es estable, la salida debería terminar siendo periódica con la misma frecuencia fundamental (4 kHz) pero más suavizada. Esto puede ser un transitorio inicial amplificado por la resonancia.

Pulso rectangular

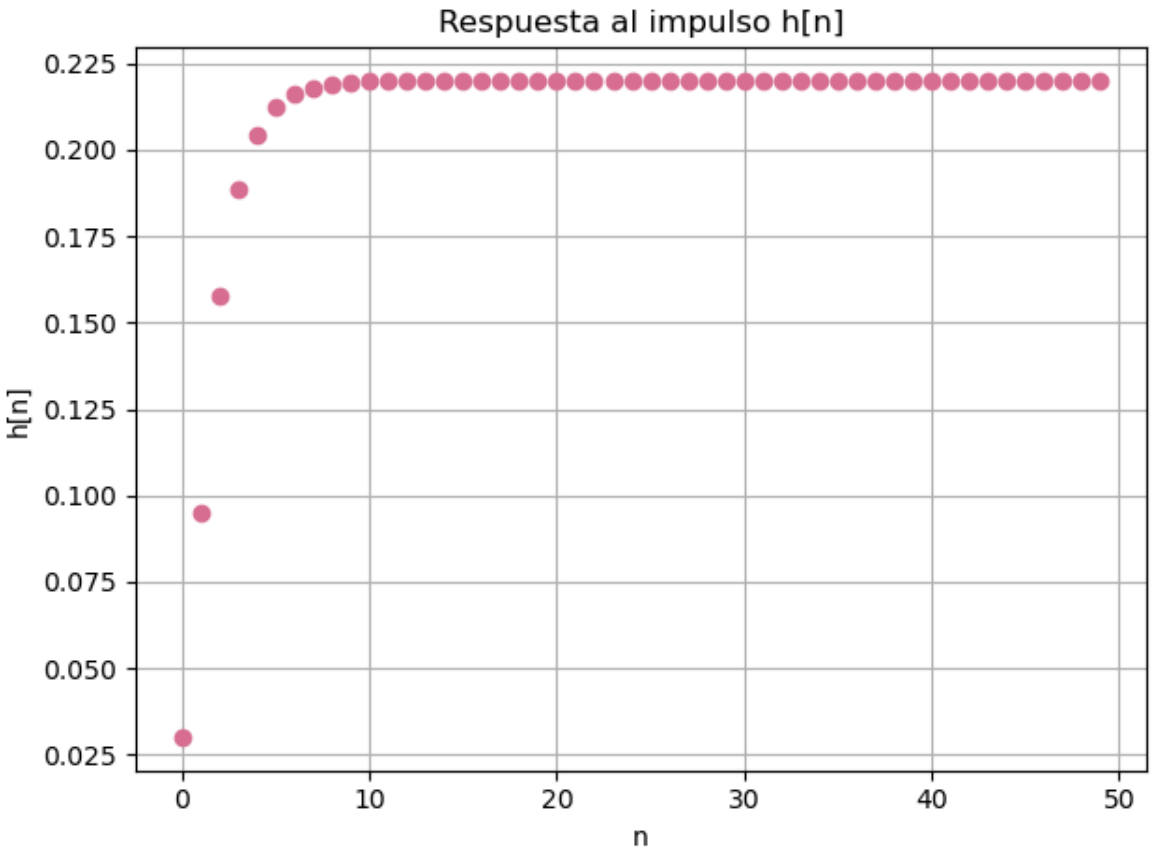
In [7]:
```
plt.figure(6)
plt.plot(tp, pulso, label="Entrada", color="darkgray")
plt.plot(tp, salidas[5][3], label="Salida", color="darkturquoise")

plt.title("Pulso")
plt.xlabel("Tiempo [s]")
plt.ylabel("Amplitud [V]")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



- Se ve un crecimiento rápido y sostenido hasta que el pulso desaparece
- La respuesta al pulso se parece a la respuesta al impulso

Respuesta al impulso

In [8]:
```
delta = np.zeros(N)
delta[0] = 1
respuestaPulso= sig.lfilter(b, a, delta)  #salida del sistema ante el impulso

plt.figure(7)
plt.plot(respuestaPulso[:50], 'o', color='palevioletred')  #respuestaPulso[:50]
plt.title("Respuesta al impulso h[n]")
plt.xlabel("n")
plt.ylabel("h[n]")
plt.grid(True)
plt.tight_layout()
plt.show()
```



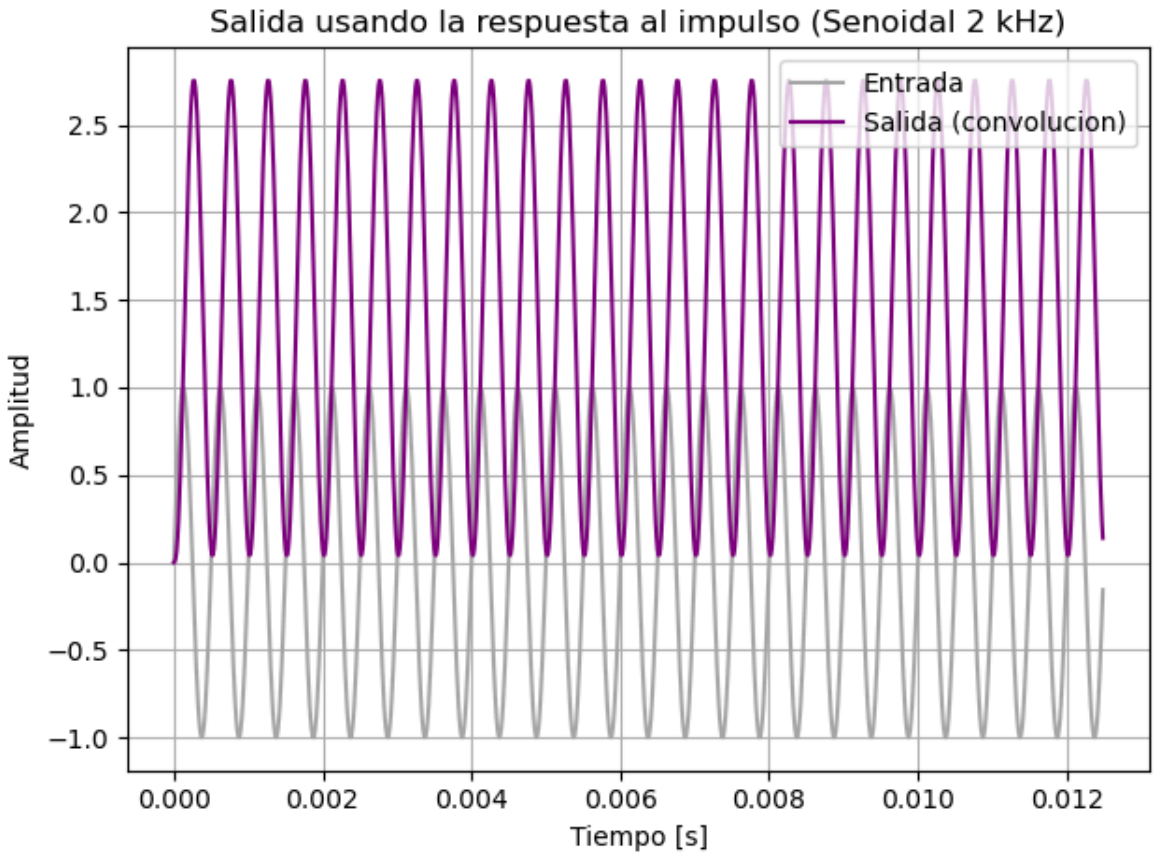
El gráfico muestra que $h[n]$ se aproxima a 0,22 y no tiende a cero, esto pasa por la presencia de un polo en $z = 1$ en la funcion tranferencia que tengo:

$$H(z) = \frac{0.03 + 0.05z^{-1} + 0.03z^{-2}}{1 - 1.5z^{-1} + 0.5z^{-2}}$$

Un polo en $z = 1$ corresponde (en el tiempo) a uan componente constane en $h[n]$

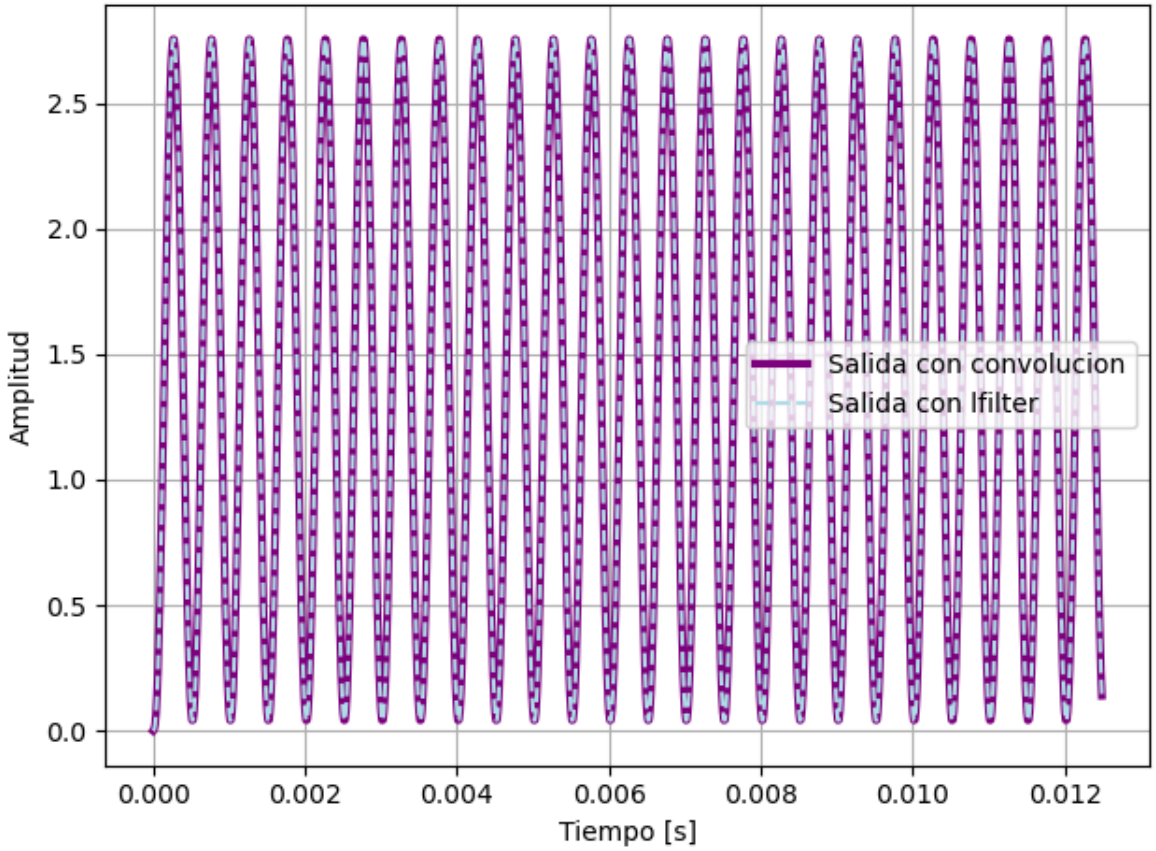
Salida con Respuesta al Impulso

In [9]:
```
conv = np.convolve(x, respuestaPulso)[:N]  #[:N] para tener el mismo tamaño

plt.figure(8)
plt.plot(t, x, label="Entrada", color="darkgray")
plt.plot(t, conv, label="Salida (convolucion)", color="purple")
plt.title("Salida usando la respuesta al impulso (Senoidal 2 kHz)")
plt.xlabel("Tiempo [s]")
plt.ylabel("Amplitud [V]")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



Mi función lfilter resuelve directamente la ecuación de diferencias, aplica la relacion recursiva entre $y[n]$ y $x[n]$, pero cuando le paso la funcion Delta se obtiene la respuesta al impulso. Los sistemas lineales tienen la propiedad de que cualquier salida se puede obtener como convolución de la entrada con el impulso. Compruebo:

In [10]:
```
plt.figure(12)
plt.plot(t, conv, label="Salida con convolucion", color="purple", linewidth=3)
plt.plot(t, salidas[0][3], label="Salida con lfilter", color="powderblue", lines
plt.xlabel("Tiempo [s]")
plt.ylabel("Amplitud [V]")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



Entonces cuando meto mi señal en una ecuacion de diferencias con lfilter tecnicamente hace lo mismo que convolucionar mi señal con la respuesta en freceuncia de la eacuacion en diferencias (conseguida con una Delta)

Ejercicio 2:

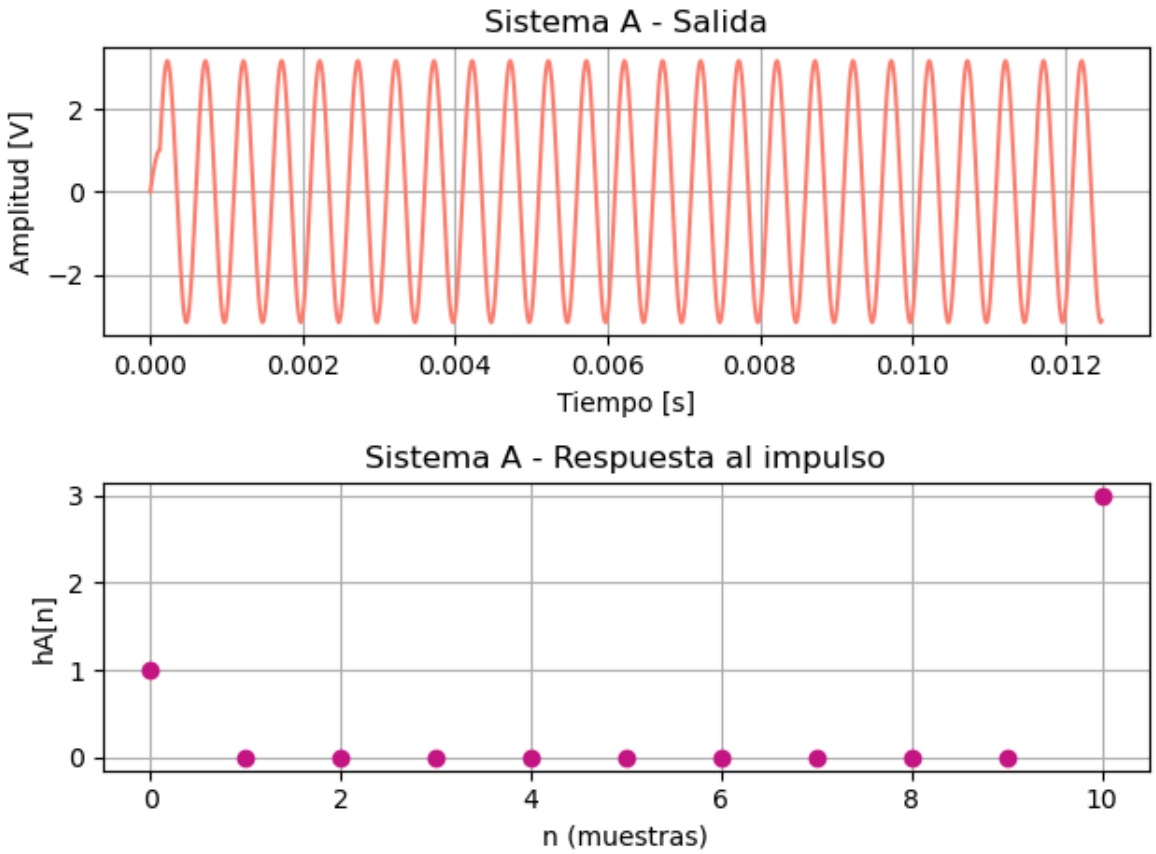
Hallar la respuesta al impulso y la salida correspondiente a una señal de entrada senoidal en los sistemas definidos mediante las siguientes ecuaciones en diferencias:

$$y[n] = x[n] + 3 \cdot x[n-10]$$

$$y[n] = x[n] + 3 \cdot y[n-10]$$

In [14]:
```
hA = np.zeros(11)
hA[0] = 1
hA[10] = (
    3  # esto esta asi porque como para la delta solo vale 1 en n=0, mi sistema
)
# la salida depende solo de la entrada, asi que al darle un pulso (delta) la res
# exactamente los coeficientes de entrada


# salida con el sen
salidaA = np.convolve(x, hA)[:N]

plt.figure(9)
plt.subplot(2, 1, 1)
plt.plot(t, salidaA, color="salmon")
plt.title("Sistema A - Salida")
```

```
plt.xlabel("Tiempo [s]")
plt.ylabel("Amplitud [V]")
plt.tight_layout()
plt.grid(True)

plt.subplot(2, 1, 2)
plt.plot(range(len(hA)), hA, "o", color="mediumvioletred")
plt.title("Sistema A - Respuesta al impulso")
plt.xlabel("n (muestras)")
plt.ylabel("hA[n]")
plt.grid(True)
plt.tight_layout()
plt.show()

# Energia de las salida
energiaA = energia(salidaA)

# Potencia de las salida
potenciaA = potencia(salidaA)

print("Energia salida A:", energiaA)
print("Potencia salida A:", potenciaA)
```



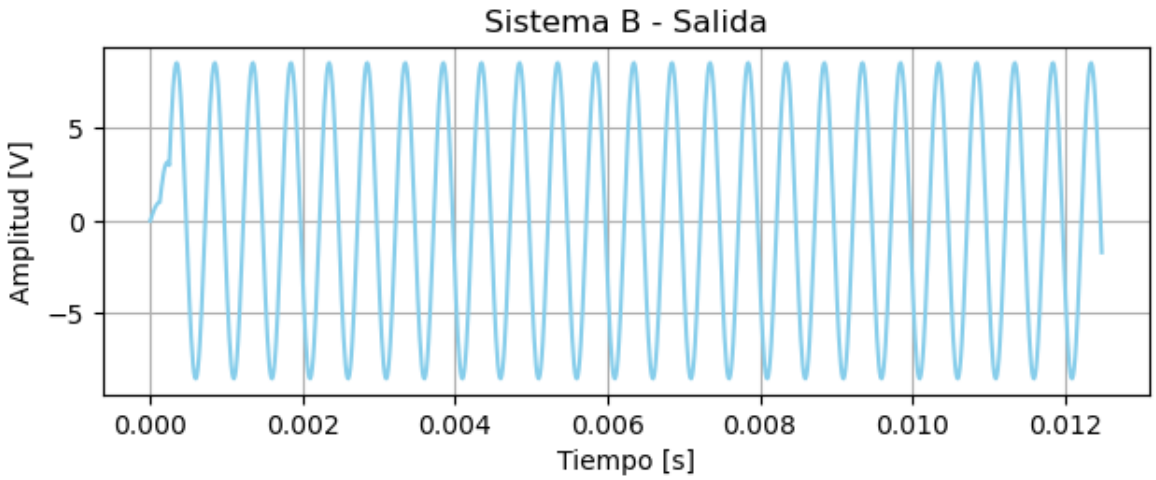

```
Energia salida A: 4969.4412545440255
Potencia salida A: 4.969441254544026
```

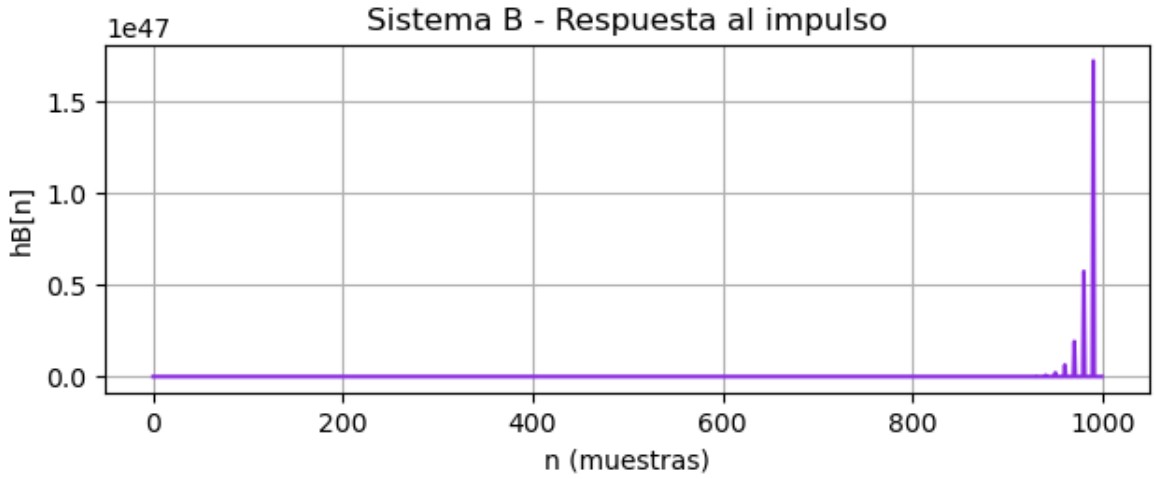
Este es un sistema FIR (es decir, no recursivo), ya que depende solo de entradas pasadas.

En su respuesta al impulso se pueden ver dos picos, uno en $n = 0$ y otro en $n = 10$, con valores 1 y 3 respectivamente. Esto indica que el sistema “recuerda” la entrada 10 muestras atrás y la amplifica.

Viendo la salida, se ve que el sistema actúa como un sumador retardado. Si la entrada es una senoidal, la salida será una combinación de la senoidal actual y otra igual pero desplazada 10 muestras, lo que genera interferencia constructiva o destructiva según la fase. Además, la salida tiene mayor energía que la entrada, lo que refleja la amplificación.

In [19]:
```
#----------Sistema B: y[n] = x[n] + 3 y[n-10]------------#
aB = np.zeros(11)
aB[0] = 1
aB[10] = -3
bB = np.array([1]) #porque no hay nadaen x

#respuesta al impulso
delta = np.zeros(N)
delta[0] = 1
hB= sig.lfilter(bB, aB, delta)
#la salida depende tambien de salidas pasadas, así que al darle un pulso (delta)
#sigue retroalimentándose y hay que calcularla con el lfilter

# salida para la senoidal
hBcortada = hB[:30]
salidaB = np.convolve(x, hBcortada)[:N] #lo corte asi no se ve como explota todo
#las lineas de codigo comentadas son como estaba originalmente (graficaba una es
#salidaB = np.convolve(x, hB)[:N]

plt.figure(10)
plt.subplot(2,1,1)
plt.plot(t, salidaB, color="skyblue")
plt.title('Sistema B - Salida')
plt.xlabel("Tiempo [s]")
plt.ylabel("Amplitud [V]")
plt.grid(True)
plt.tight_layout()
plt.show()

plt.subplot(2,1,2)
plt.plot(hB, label='respuesta al impulso', color="blueviolet")
plt.title('Sistema B - Respuesta al impulso')
plt.xlabel('n (muestras)')
plt.ylabel('hB[n]')
plt.grid(True)
plt.tight_layout()
plt.show()

#Energia de las salida
energiaB = energia(salidaB)

# Potencia de las salidas
potenciaB = potencia(salidaB)

print("Energia salida B:", energiaB)
print("Potencia salida B:", potenciaB)
```



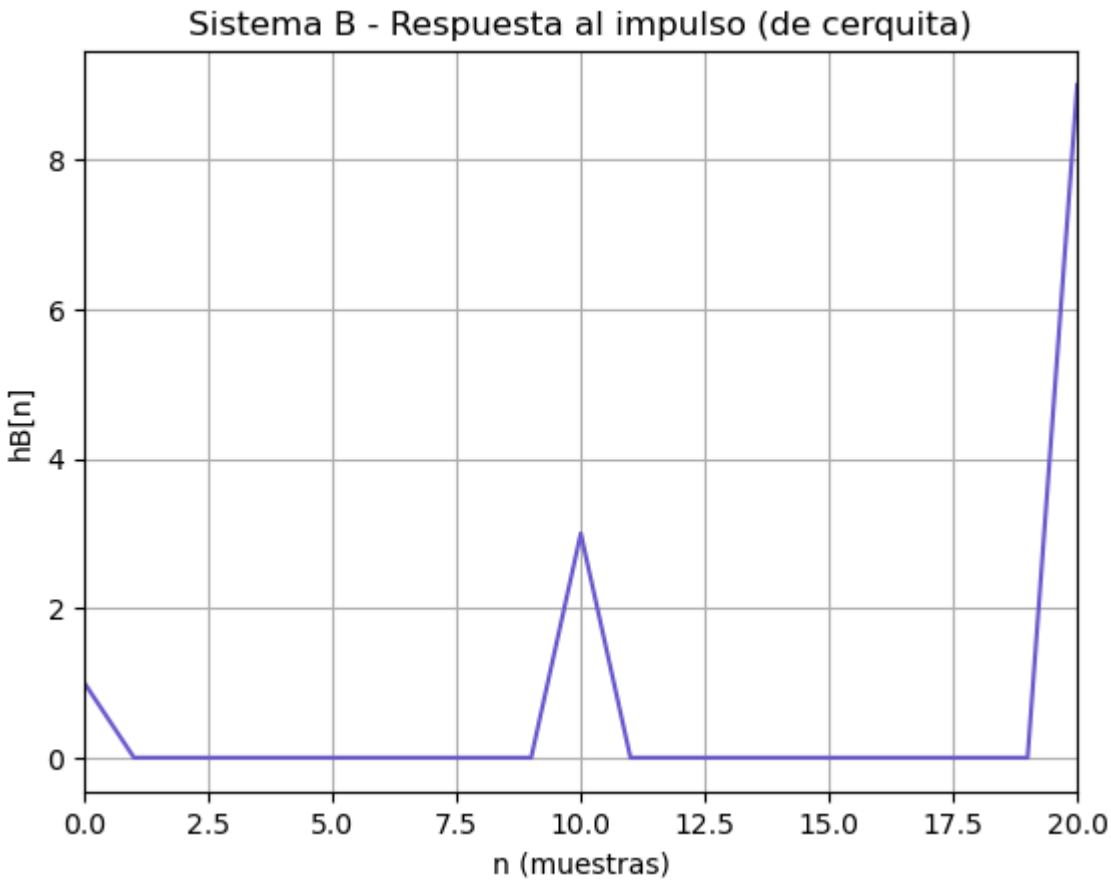



```
Energia salida B: 35839.44125454403
Potencia salida B: 35.83944125454403
```

Este es un sistema IIR (es decir, recursivo), ya que depende solo de salidas pasadas.

En la respuesta al impulso se puede cer como gracias a la retroalimentación la respuesta crece exponencialmente (cada 10 muestras), para muestras altas se alcanzan valores altísimos, si cortamos la gráfica para estudiar las muestras mas "tempranas":

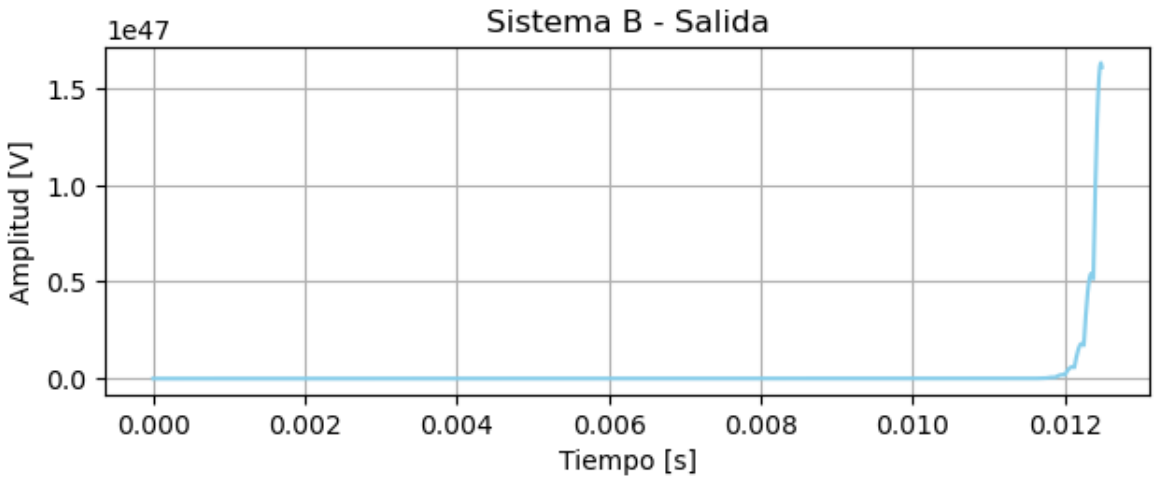
In [22]:
```
plt.figure(13)
plt.plot(hB[:21], label='respuesta al impulso', color="slateblue")
plt.title('Sistema B - Respuesta al impulso (de cerquita)')
plt.xlabel('n (muestras)')
plt.ylabel('hB[n]')
plt.grid(True)
plt.xlim(0, 20)
plt.show()
```



- Se puede ver como en el valor inicial, al igual que en la función Delta, este es igual a 1
- También se puede cer como la retroalimentación empieza en n=10

En la salida sucede lo mismo el sistema no es estable, si le quitamos el "zoom" agregado para ver la señal de forma uniforme, esta quedaría asi:

In [23]:
```
# salida para la senoidal
salidaB = np.convolve(x, hB)[:N]

plt.figure(14)
plt.subplot(2,1,1)
plt.plot(t, salidaB, color="skyblue")
plt.title('Sistema B - Salida')
plt.xlabel("Tiempo [s]")
plt.ylabel("Amplitud [V]")
plt.grid(True)
plt.tight_layout()
plt.show()
```



Cada valor de salida se amplifica y se propaga hacia adelante, la energía de la salida crece sin límites.

Bonus

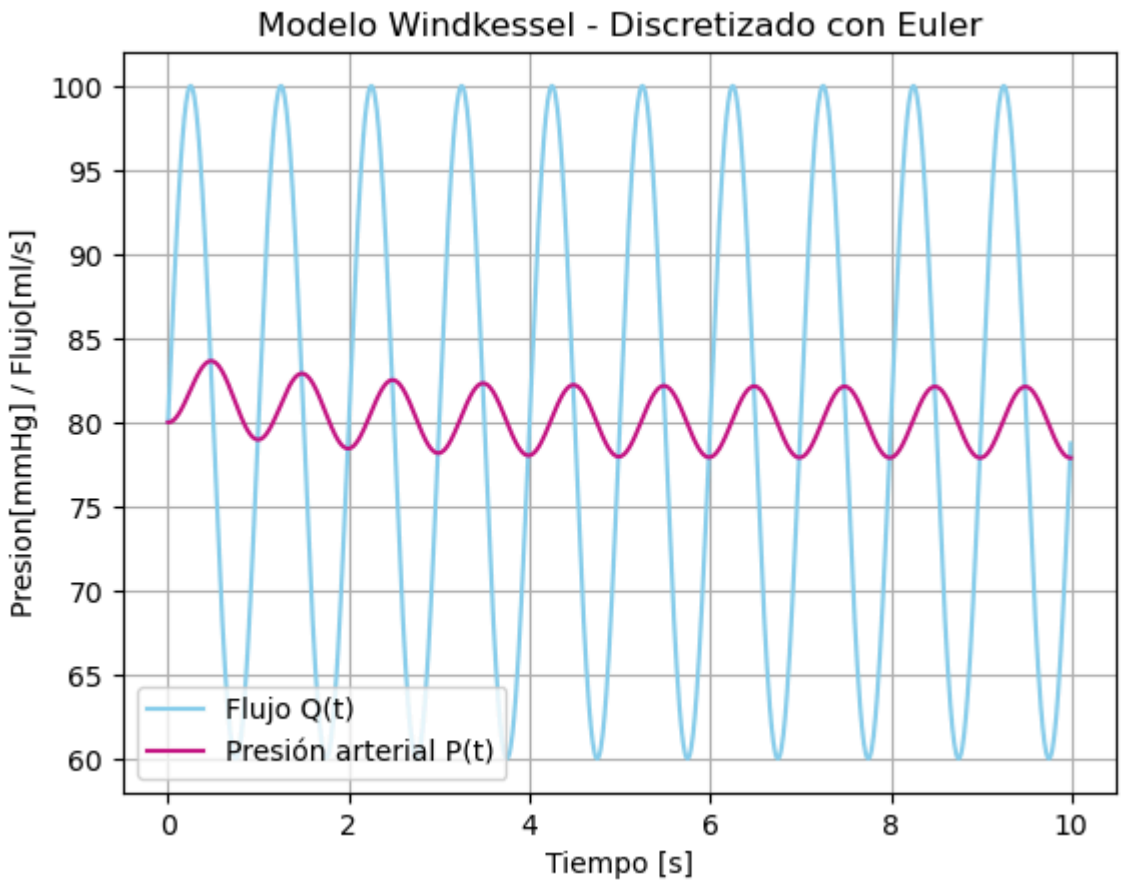
Discretizar la siguiente ecuación diferencial correspondiente al modelo de Windkessel que describe la dinámica presión-flujo del sistema sanguíneo:

$$C\frac{dP(t)}{dt} + \frac{P(t)}{R} = Q(t)$$

Considere valores típicos de Compliance y Resistencia vascular

In [24]:
```
# Parámetros de la senoidal (flujo)
Qn = 20   # amplitud de la onda de flujo ml/s
dc = 80   # flujo medio equivalente a presión base en mmHg
ff = 1     #frecuencia Hz
ph = 0       #fase inicial
fs = 100     #frecuencia de muestreo Hz
C = 1.5     # ml/mmHg
R = 1.0     # mmHg·s/ml
dt = 1/fs   # paso temporal consistente con la señal

t, Q = sen(Qn, dc, ff, ph, N, fs)

P = np.zeros(N) #es un array donde almaceno la presion arterial en cada instante
P[0] = 80  # (valor típico de presión sistólica inicial o presión de referencia)

#metodo de Euler para discretizar la ecuacion diferencial
for n in range(N-1):#itero de la muestra 0 hasta la penúltima N-1
    P[n+1] = P[n] + dt*(Q[n] - P[n]/R)/C

#en cada paso la presion aumenta si el flujo Q[n] es mayor que P[n]/R, y disminu

plt.figure(11)
plt.plot(t, Q, label='Flujo Q(t)', color='skyblue')
#Q es la entrada del sistema, simula el flujo sanguineo pulsatil (un latido por
plt.plot(t, P, label='Presión arterial P(t)', color='mediumvioletred')
#P es la salida, es la presion arterial que responde al flujo
plt.xlabel('Tiempo [s]')
```

```
plt.ylabel('Presion[mmHg] / Flujo[ml/s]')
plt.title('Modelo Windkessel - Discretizado con Euler')
plt.legend()
plt.grid(True)
plt.show()
```



Se aplicó el método de Euler explícito para aproximar la evolución de la presión arterial $P[n]$ en función del flujo pulsátil $Q[n]$, con frecuencia de muestreo f_s=100 Hz.

- La señal de flujo $Q(t)$ simula un latido cardíaco senoidal con componente continua (flujo medio)
- La forma de P(t) refleja la capacidad del sistema vascular de almacenar y liberar volumen (complianza), y disipar energía (resistencia) La implementación con Euler permite observar cómo la resistencia y la complianza modulan la forma de la señal de presión, revelando el rol del sistema vascular como amortiguador hidráulico.

Conclusión:

Se logró entender como funciona una ecuacion en diferencias y como me altera la señal de entrada, tambien se logro aprénder sobre los tipos de ecuación y como las recursividades pueden volver una salida inestable.

Autoevaluación del aprendizaje

Resolví la tarea pero me costó

Creo que realmente no tenia mucha idea de que eran estos temas y como funcionaban, pero luego de indagar logre comprender los conceptos del trabajo. Respecto al uso de la IA, se ultilizo para problemas con python como siempre y para poder escribir en Latex.

In [ ]: