

# Separación de Fuentes: Beamforming

# Modelo de las Señales Capturadas

- Se asume que hay una o varias señales de interés (SOI), con direcciones de arribo conocidas, tal que:

$$\mathbf{A}_f = \begin{bmatrix} e^{-i2\pi f T_{1:1}} & e^{-i2\pi f T_{1:2}} & \dots & e^{-i2\pi f T_{1:D}} \\ e^{-i2\pi f T_{2:1}} & e^{-i2\pi f T_{2:2}} & \dots & e^{-i2\pi f T_{2:D}} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-i2\pi f T_{M:1}} & e^{-i2\pi f T_{M:2}} & \dots & e^{-i2\pi f T_{M:D}} \end{bmatrix} \quad \mathbf{S}_f = \begin{bmatrix} s_1(f) \\ s_2(f) \\ \vdots \\ s_D(f) \end{bmatrix}$$

$$\mathbf{X}_f = \mathbf{A}_f \mathbf{S}_f$$

Donde:

$s_d$ : señal de origen "d"

N: tamaño de la señal (o de la ventana de la señal)

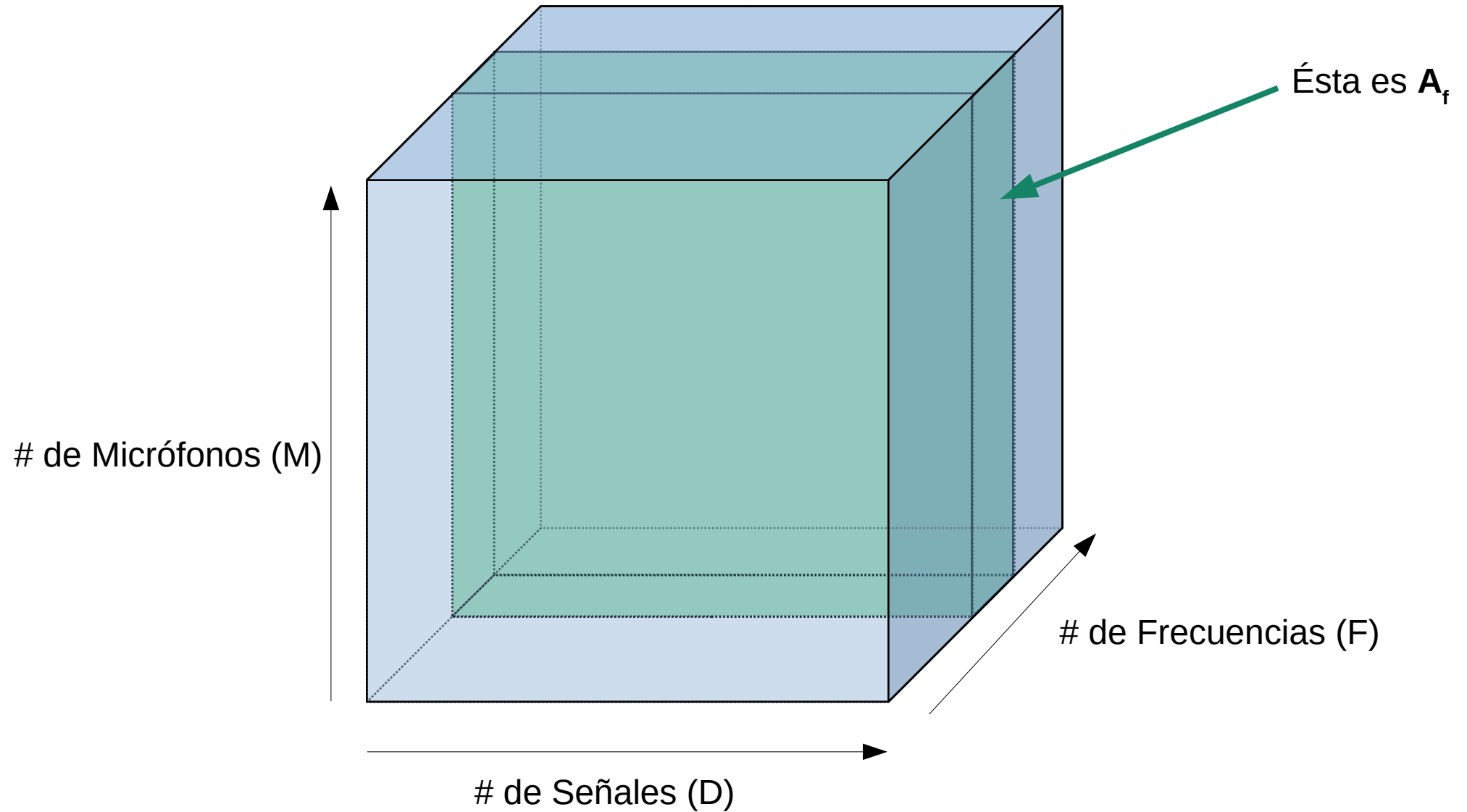
$T_{m:d}$ : retraso recibido de la señal  $s_d$  en el micrófono m

$\mathbf{A}_f$ : matriz que contiene los vectores de dirección *para la frecuencia f*

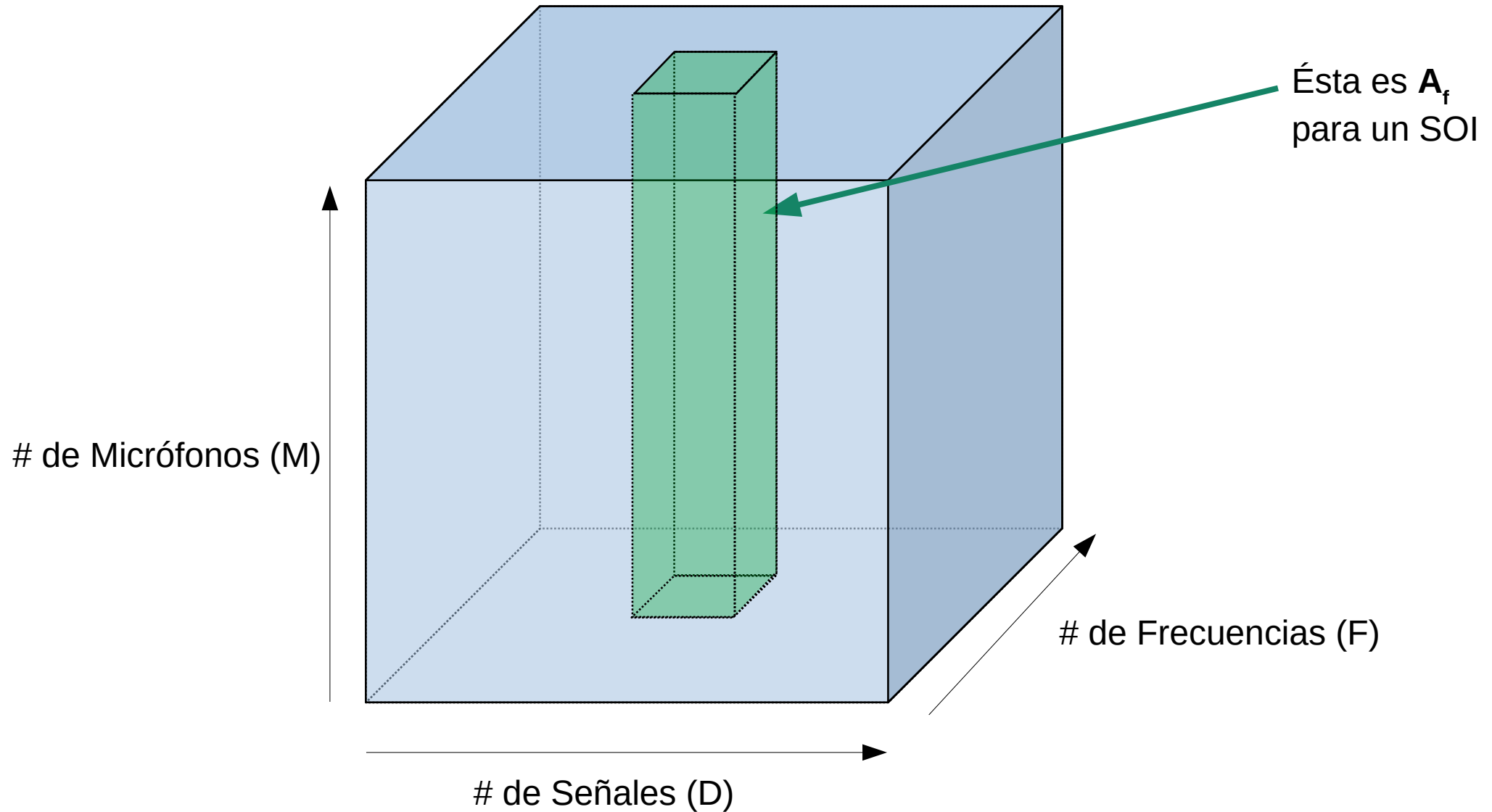
$\mathbf{X}_f$ : señales capturadas *en la frecuencia f*; cada renglón representa un micrófono

$\mathbf{S}_f$ : señales de origen *en la frecuencia f*; cada renglón representa una señal de origen

# RECORDATORIO



# Para cada SOI, Para cada Frecuencia



# Objetivo

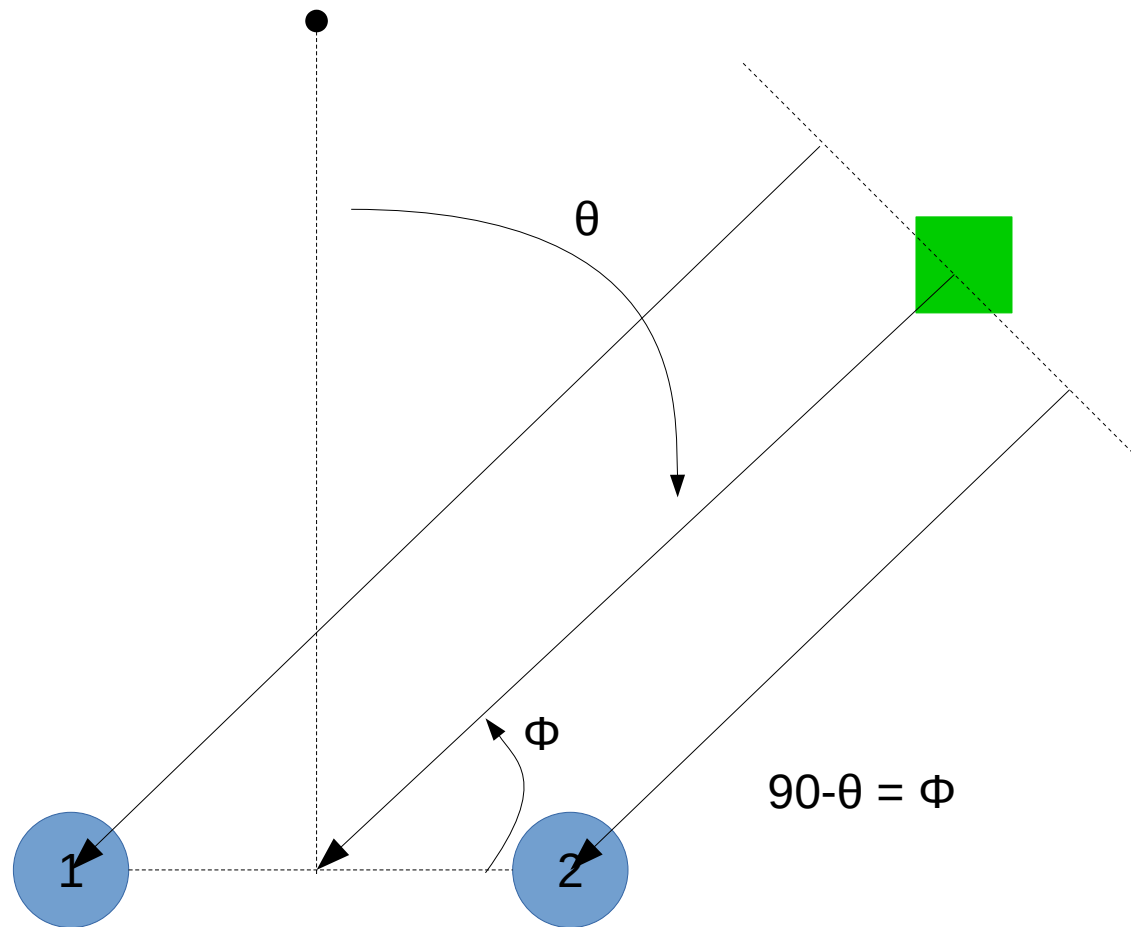
- Estimar las señales en  $\mathbf{S}$  por medio de aplicar una matriz adicional  $\mathbf{W}$  a  $\mathbf{X}$ .

$$\hat{\mathbf{S}}_f = \mathbf{W}_f \mathbf{X}_f$$

# Relación entre $A$ y $W$

- Es muy tentador decir que  $W = A^{-1}$ .
- Y la solución es bastante cercana a ello.
- Pero, es importante primero saber qué significa llevar a cabo Beamforming.

# Beamforming



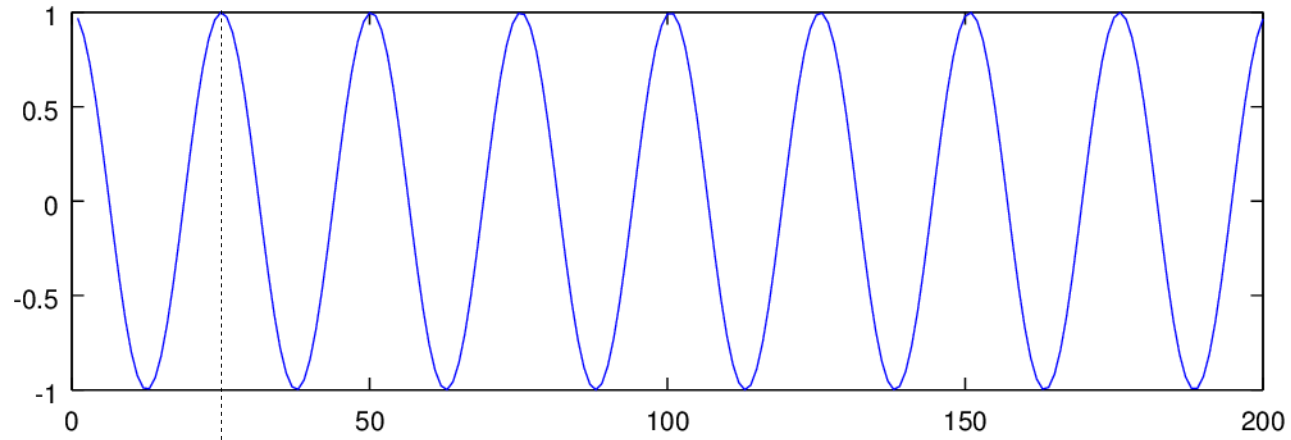
# Beamforming

- En esta parte del procesamiento, asumimos que ya conocemos el DOA de la fuente.
- Por lo tanto, también ya también conocemos el desfase que necesitamos llevar a cabo en una de las señales para que “se parezcan lo más posible”.

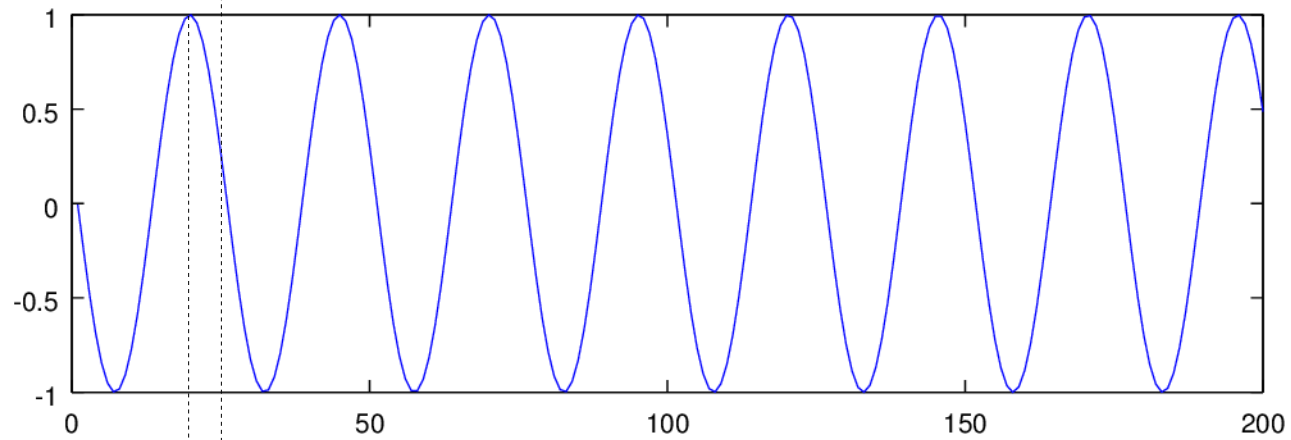


# Desfase

Señal 1



Señal 2

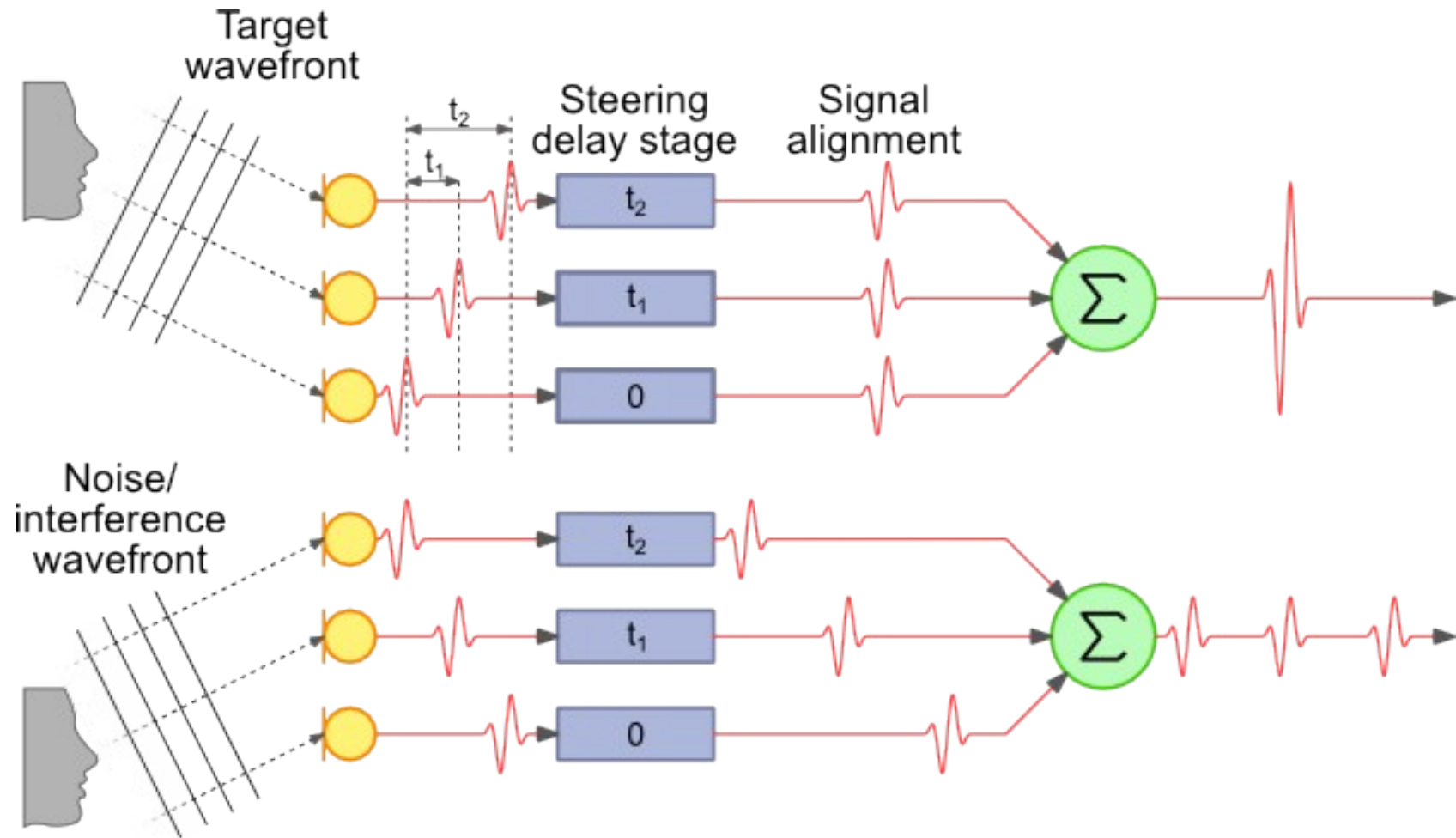


$t_{2-1}$

# Desfase

- ¿Qué sucedería si desfasáramos la segunda señal acorde a éste DOA y luego sumáramos las dos señales (después de desfasar)?
- ¿Qué sucedería si hubiera una segunda fuente *interferente*?

# Delay-and-Sum Beamforming



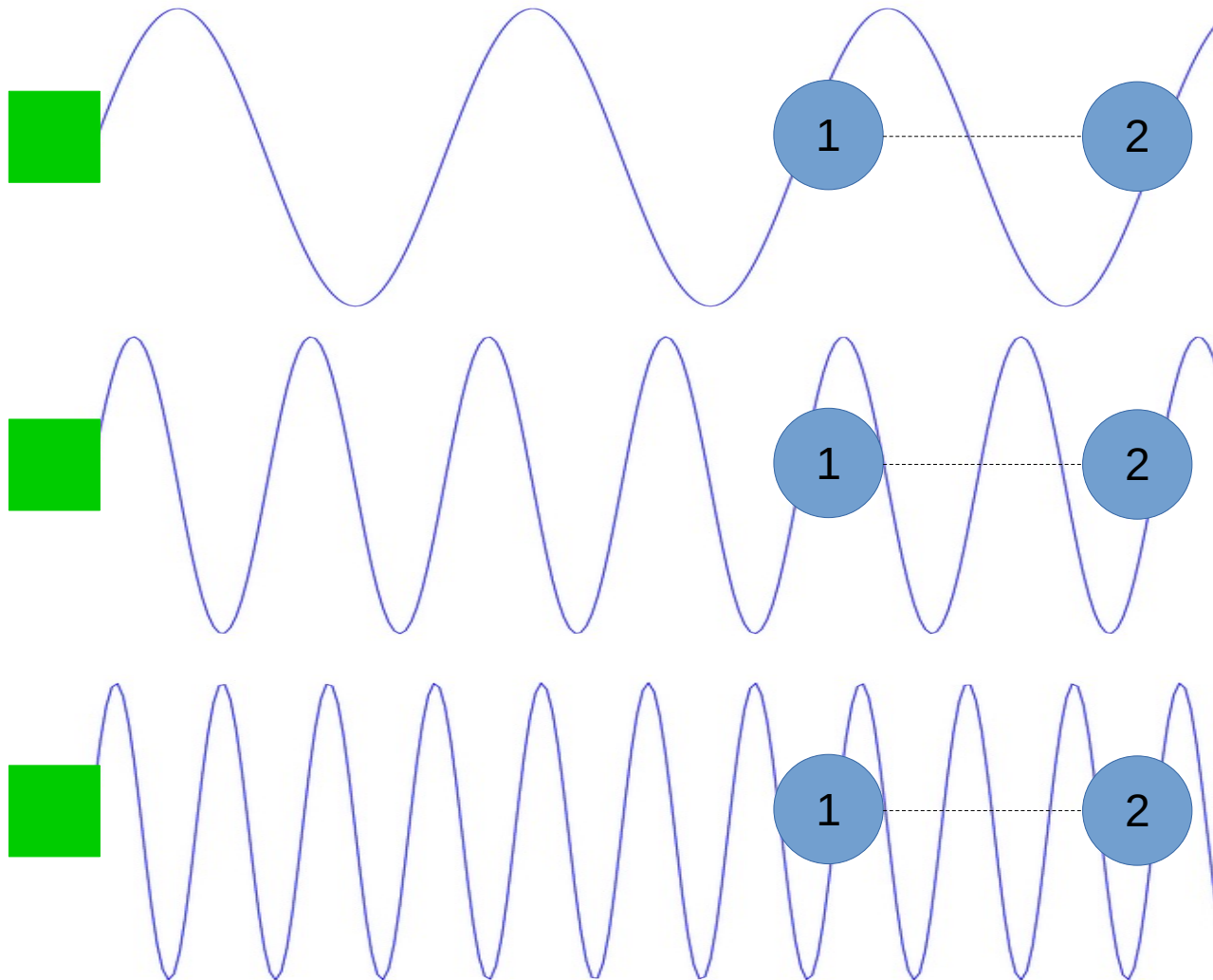
# Delay-and-Sum Beamforming

- Se calcula el desfase apropiado al DOA por cada micrófono.
- Se desfasa cada señal.
- Se suma punto a punto, y la suma se divide entre el número de micrófonos.
- El resultado se saca a la salida.

# Distancia Inter-micrófono vs Frecuencia de la Señal

- Beamforming intenta “recrear” la señal en la dirección de interés.
- Pero, si no “cabe” entre los micrófonos, no podrá hacerlo.
- Es decir:

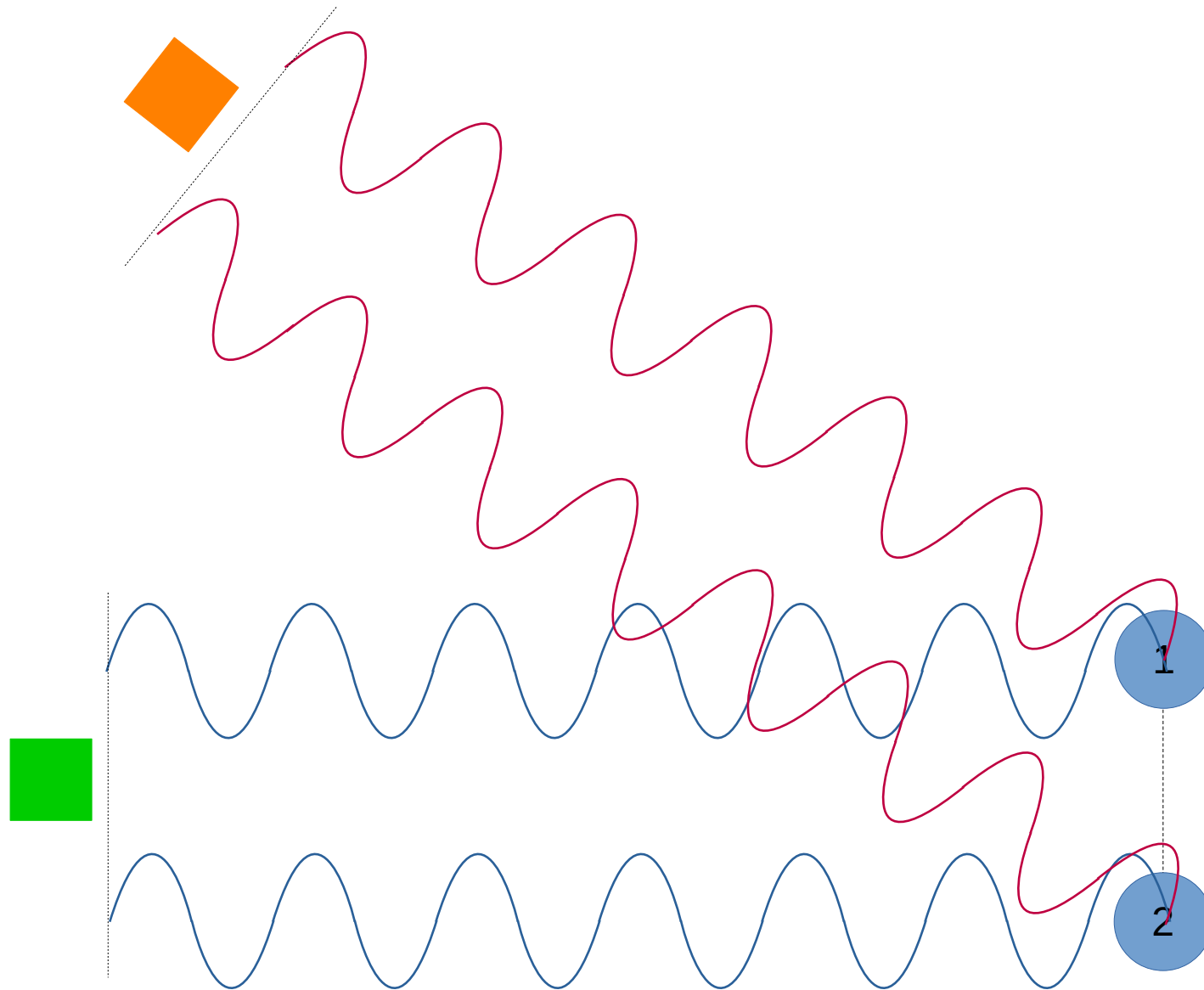
# Distancia Inter-micrófono vs Frecuencia de la Señal



# Distancia Inter-micrófono vs Frecuencia de la Señal

- Por otro lado, algunas frecuencias llegarán al mismo tiempo (o “en fase”) a los micrófonos aún cuando están en diferentes lugares.
  - Entre más alta la frecuencia, más posibilidad de que esto suceda.
- Es decir:

# Distancia Inter-micrófono vs Frecuencia de la Señal





# Distancia Inter-micrófono vs Frecuencia de la Señal

- Esto se conoce como: *spatial aliasing*.
  - No he encontrado una traducción al español buena.
    - “Solapamiento espacial” es la más popular.
  - **ADVERTENCIA:** *aliasing* se utiliza para otros tipos de distorsión (como cuando se viola el criterio de Nyquist).

# Distancia Inter-micrófono vs Frecuencia de la Señal

- Entonces, dependiendo de la distancia entre los micrófonos:
  - Bajas frecuencias pueden ser distorsionadas.
    - Voces bajas suenan “robóticas”.
  - Altas frecuencias tienen más posibilidad de llegar a “en fase” en direcciones que no son de interés.
    - Y no serán reducidas en magnitud.

# Distancia Inter-micrófono vs Frecuencia de la Señal

- Selección de la distancia inter-micrófono es un acto de balance:
  - Distancias pequeñas:
    - PRO: frecuencias altas removidas en direcciones interferentes, y las fuentes pueden estar más cercanas a los micrófonos
    - CON: frecuencias bajas distorsionadas.
  - Distancias grandes: viceversa.

# Distancia Inter-micrófono vs Frecuencia de la Señal

- Si:  $f = \frac{c}{\lambda_f}$ 
  - Donde  $f$  es la frecuencia,  $c$  es la velocidad de propagación (alrededor de 343 m/s en el aire), y  $\lambda_f$  es la longitud de onda de la frecuencia  $f$ .
- Para evitar corromper bajas frecuencias:  $d > \lambda_{f_{min}}$ 
  - Donde  $d$  es la distancia entre micrófonos.
- Para evitar *spatial aliasing*:  $d < \frac{\lambda_{f_{max}}}{2}$
- **Buen balance:** 0.18 y 0.21 m para señales de voz.
  - Parecido a la distancia entre orejas de humanos.

# Delay-and-Sum Beamforming

- DAS es la base de otros filtros espaciales, por lo que estas consideraciones son importantes de tomar en cuenta aún cuando no se está utilizando este filtro espacial.

# En el Dominio de la Frecuencia

- Recuerden que el desfase se puede también llevar a cabo por medio de una multiplicación de una exponencial compleja en el dominio de la frecuencia:

$$g(t - T) = F^{-1}(G(t) e^{-i2\pi f T})$$

- Pero se tiene que multiplicar a TODAS las frecuencias por el exponencial adecuado.

# Delay-and-Sum Beamforming (Frecuencia)

- Calcular las exponenciales adecuadas para cada frecuencia de la señal.
  - Esto es el “steering vector” del arreglo:  $\mathbf{W}_f$ .
- Hacerle FFT a las señales de entrada.
- Aplicar: 
$$\hat{\mathbf{S}}_f = \mathbf{W}_f^H \mathbf{X}_f$$
- Sacarle la IFFT a  $\hat{\mathbf{S}}_f$ , el cual es la salida.

$^H$ : es la operación de transposición conjugada de una matriz, también conocida como la *conjugación Hermitiana*.

# W en Delay-and-Sum (DAS)

$$\hat{\mathbf{S}} = \mathbf{W}^H \mathbf{X}$$

$$\mathbf{X} = \begin{bmatrix} x_1(f_1) & x_1(f_2) & \cdots & x_1(f_N) \\ x_2(f_1) & x_2(f_2) & \cdots & x_2(f_N) \\ \vdots & \vdots & \ddots & \vdots \\ x_M(f_1) & x_M(f_2) & \cdots & x_M(f_N) \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ e^{-i2\pi f_1 T_2} & e^{-i2\pi f_2 T_2} & \cdots & e^{-i2\pi f_N T_2} \\ e^{-i2\pi f_1 T_3} & e^{-i2\pi f_2 T_3} & \cdots & e^{-i2\pi f_N T_3} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-i2\pi f_1 T_M} & e^{-i2\pi f_2 T_M} & \cdots & e^{-i2\pi f_N T_M} \end{bmatrix}$$

Donde:

$f_n$ : es la frecuencia n

$T_m$ : es el desfase que se observa de la dirección de interés para micrófono m

X: matriz de las señales capturadas, transformadas con FFT

**W: una matriz que contiene como columnas a los *steering vector* ( $\mathbf{A}_f$ )**



# W en Delay-and-Sum (DAS)

$$\hat{\mathbf{S}} = \mathbf{W}^H \mathbf{X}$$

$$\mathbf{X} = \begin{bmatrix} x_1(f_1) & x_1(f_2) & \cdots & x_1(f_N) \\ x_2(f_1) & x_2(f_2) & \cdots & x_2(f_N) \\ \vdots & \vdots & \ddots & \vdots \\ x_M(f_1) & x_M(f_2) & \cdots & x_M(f_N) \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ e^{-i2\pi f_1 T_2} & e^{-i2\pi f_2 T_2} & \cdots & e^{-i2\pi f_N T_2} \\ e^{-i2\pi f_1 T_3} & e^{-i2\pi f_2 T_3} & \cdots & e^{-i2\pi f_N T_3} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-i2\pi f_1 T_M} & e^{-i2\pi f_2 T_M} & \cdots & e^{-i2\pi f_N T_M} \end{bmatrix}$$

Cada columna de  $\mathbf{W}$  es un  $\mathbf{A}_f$  para una SOI

Donde:

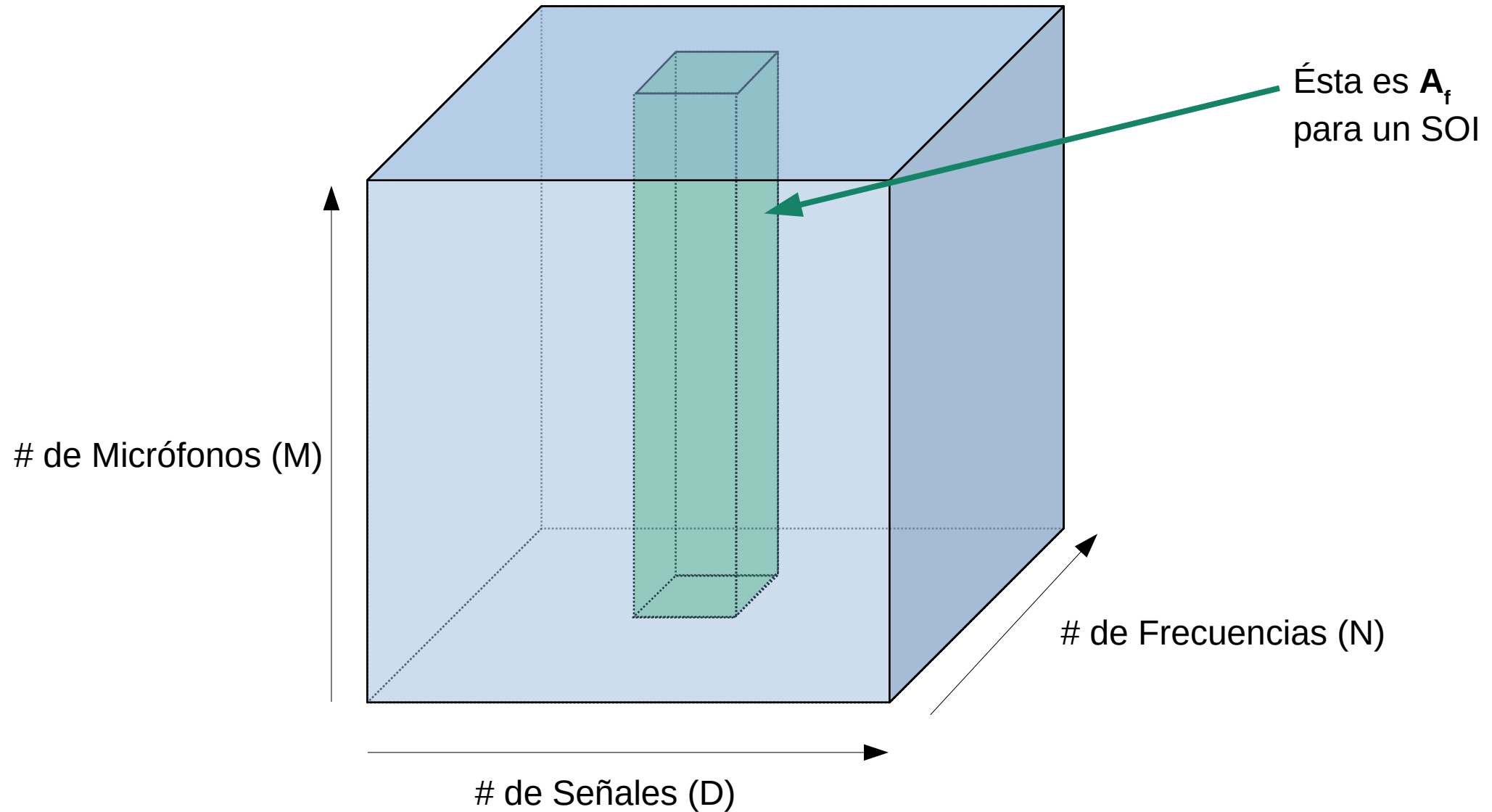
$f_n$ : es la frecuencia n

$T_m$ : es el desfase que se observa de la dirección de interés para micrófono m

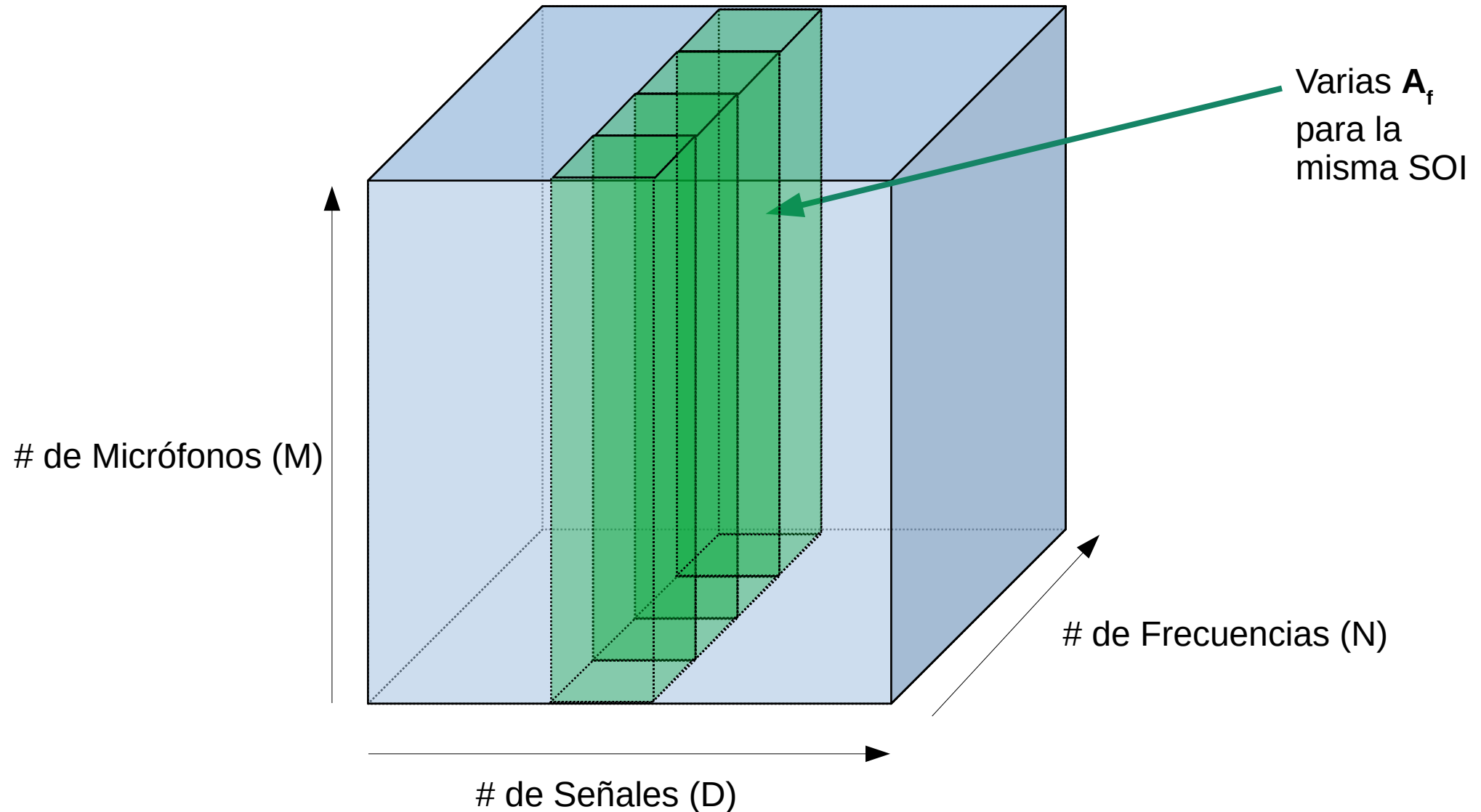
$\mathbf{X}$ : matriz de las señales capturadas, transformadas con FFT

$\mathbf{W}$ : una matriz que contiene como columnas a los *steering vector* ( $\mathbf{A}_f$ )

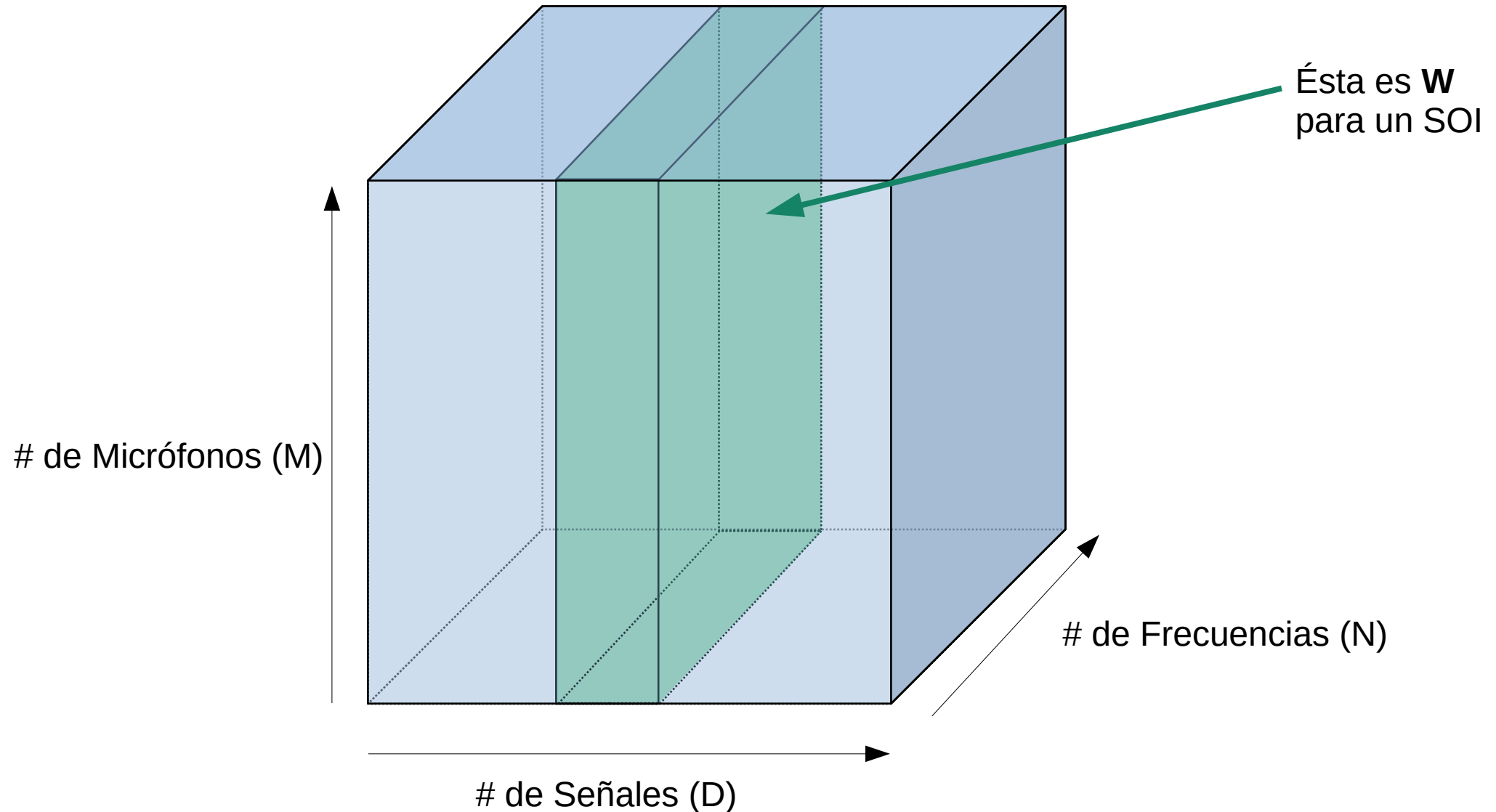
# Relación entre $A$ y $W$ : se parecen, pero no son lo mismo



# Relación entre $A$ y $W$ : se parecen, pero no son lo mismo



# Relación entre A y W: se parecen, pero no son lo mismo



# W en Delay-and-Sum (DAS)

$$\hat{\mathbf{S}} = \mathbf{W}^H \mathbf{X}$$

$$\mathbf{X} = \begin{bmatrix} x_1(f_1) & x_1(f_2) & \cdots & x_1(f_N) \\ x_2(f_1) & x_2(f_2) & \cdots & x_2(f_N) \\ \vdots & \vdots & \ddots & \vdots \\ x_M(f_1) & x_M(f_2) & \cdots & x_M(f_N) \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ e^{-i2\pi f_1 T_2} & e^{-i2\pi f_2 T_2} & \cdots & e^{-i2\pi f_N T_2} \\ e^{-i2\pi f_1 T_3} & e^{-i2\pi f_2 T_3} & \cdots & e^{-i2\pi f_N T_3} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-i2\pi f_1 T_M} & e^{-i2\pi f_2 T_M} & \cdots & e^{-i2\pi f_N T_M} \end{bmatrix}$$

Cada columna de  $\mathbf{W}$  es un  $\mathbf{A}_f$  para una SOI

Donde:

$f_n$ : es la frecuencia n

$T_m$ : es el desfase que se observa de la dirección de interés para micrófono m

X: matriz de las señales capturadas, transformadas con FFT

**W**: una matriz que contiene como columnas a los *steering vector* ( $\mathbf{A}_f$ )

# W en Delay-and-Sum (DAS)

$$\hat{S} = W^H X$$

¿Y esto?

$$X = \begin{bmatrix} x_1(f_1) & x_1(f_2) & \cdots & x_1(f_N) \\ x_2(f_1) & x_2(f_2) & \cdots & x_2(f_N) \\ \vdots & \vdots & \ddots & \vdots \\ x_M(f_1) & x_M(f_2) & \cdots & x_M(f_N) \end{bmatrix}$$

$$W = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ e^{-i2\pi f_1 T_2} & e^{-i2\pi f_2 T_2} & \cdots & e^{-i2\pi f_N T_2} \\ e^{-i2\pi f_1 T_3} & e^{-i2\pi f_2 T_3} & \cdots & e^{-i2\pi f_N T_3} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-i2\pi f_1 T_M} & e^{-i2\pi f_2 T_M} & \cdots & e^{-i2\pi f_N T_M} \end{bmatrix}$$

Donde:

$f_n$ : es la frecuencia n

$T_m$ : es el desfase que se observa de la dirección de interés para micrófono m

X: matriz de las señales capturadas, transformadas con FFT

**W: una matriz que contiene como columnas a los *steering vector* ( $A_f$ )**

# Efecto de la Hermitiana

- Al aplicar la transpuesta conjugada, la parte imaginaria de un número complejo se niega.
- Para el caso de un desfase en el dominio de la frecuencia, esto resulta en “negar” el efecto de desfase observado para la dirección de interés.

$$\mathbf{W}_{f_1}^H = \begin{bmatrix} 1 \\ e^{-i2\pi f_1 T_2} \\ e^{-i2\pi f_1 T_3} \end{bmatrix}^H = \begin{bmatrix} 1 & e^{i2\pi f_1 T_2} & e^{i2\pi f_1 T_3} \end{bmatrix}$$

# ¿Qué es un steering vector?

- Es el vector que “dirige” la atención del filtro a una dirección de interés.
  - Esto difiere del *direction vector* que es más descriptivo de los desfases que sufre las señales capturadas dada una dirección.
- PERO, en el caso de Delay-and-Sum, y por el efecto de la Hermitiana: el steering vector y el direction vector son equivalentes.
- Esto no va a ser el caso en las siguientes técnicas.



# Ejercicio #1

- Descarguen:
  - das.m
  - delay\_f.m
  - trianglewave.m

# Ejercicio #1

- El script `das` crea dos señales de origen (una senoidal y otra triangular) y simula la entrada en varios micrófonos en un arreglo lineal.
- Luego aplica el Delay-and-Sum Beamforming en una dirección (en radianes) deseada definida en:

`doa_steer`

# Ejercicio #1

- Crea tres figuras:
  - Figura 1: las señales de origen.
  - Figure 2: la señal capturadas en el primer micrófono.
  - Figure 3: señales de origen superpuestas sobre el resultado de salida del beamforming.

# Ejercicio #1

- También pueden cambiar:
  - M: número de micrófonos utilizados
  - d: distancia entre los micrófonos en metros

# OJO: W y Hermitiana

- La implementación de la transpuesta en Octave lleva a cabo la conjugación compuesta (Hermitiana) de cada elemento del vector si es de número complejos.
  - Por lo tanto, este código entrega el resultado esperado.
- Para hacer una transpuesta común y corriente con números complejos en Octave se hace así:

`w.'`

con un punto antes de la comilla

# OJO: W y Hermitiana

- Si se cambian las líneas que crean a los steering vectors W (líneas 43-49), tal que los exponenciales tengan el signo invertido al que se utiliza en el script delay\_f.
- Es decir, así:  
    delay\_f.m:  $\exp(-i*2*\pi*w(f)*time)$   
    das.m:  $\exp(i*(2*\pi*w(f)*m*d/c)*\sin(doa\_steer))$
- Y se utiliza la transpuesta normal en:  
     $o\_f(f) = w\_c(:,f) \cdot X(:,f);$
- das entrega el mismo resultado.

# Ejercicio #1

- ¿Qué sucede si reducimos el número de micrófonos?

$$M = 3$$

# Ejercicio #1

- ¿Qué sucede si reducimos el número de micrófonos?

$$M = 3$$

- La calidad de la salida se reduce.
  - Recuerden que la señal de interferencia se reduce entre el número de micrófonos.
  - Entre más micrófonos, más calidad en la SOI.



# Ejercicio #1

- Regresa al número de micrófonos original: 8.
- ¿Qué sucede si modificamos a `doa_steer` a apuntar a la otra señal de origen?

# Ejercicio #1

- Regresa al número de micrófonos original: 8.
- ¿Qué sucede si modificamos a `doa_steer` a apuntar a la otra señal de origen?
  
- La salida no se parece en absoluto a la segunda señal de origen.

# Ejercicio #1

- ¿Qué sucede si aumentamos considerablemente el número de micrófonos?  
M = 32 (cuatro veces más micrófonos)

# Ejercicio #1

- ¿Qué sucede si aumentamos considerablemente el número de micrófonos?  
 $M = 32$  (cuatro veces más micrófonos)
- La salida se mejora considerablemente.
  - La interferencia tiene más energía que la SOI.
    - Se requiere más información para recrearla.
  - Aunque todavía no es perfecta.

# Conclusiones de Delay-and-Sum

- Fácil de implementar.
- Pero requiere muchos micrófonos para funcionar bien.
- No funciona bien cuando las interferencias tienen más energía que la SOI.
  - Mejor dicho, cuando la razón de Señal a Interferencia (SIR) es baja.

# Notas de Delay-and-Sum

- Se puede generalizar a arreglos de micrófonos de más de una dimensión.
  - Sólo se tienen que manejar bien los desfases.

# Steering vector vs. Direction Vector

- El *steering vector* ( $W$ ) “dirige” la atención del filtro a una dirección de interés.
- El *direction vector* ( $A$ ) describe los desfases que sufren las señales capturadas dada una dirección.
- En el caso de Delay-and-Sum, y por el efecto de la Hermitiana: el steering vector y el direction vector son equivalentes.
- **PERO**, esto no es el caso con otras técnicas.

# Otras Formas de Beamforming

- Minimum Variance Distortionless Response
- Linearly Constrained Minimum Variance
- Generalized Sidelobe Canceller
- Frequency Masking \*\*\*

\*\*\* realmente no es beamforming... larga historia...



# Minimum Variance Distortionless Response (MVDR)

# Minimum Variance Distortionless Response

- Historia triste:
  - Es también conocido como “Capon Beamformer”, ya que se le adjudica a Capon su presentación inicial en 1969.
  - Pero, realmente lo presentó originalmente Levin en 1964 para señales sísmicas.
    - Capon después lo reformula en 1967 para hacerlo más eficiente, y en 1969 lo presenta para otras áreas que no son Sísmicas, como Telecomunicaciones.
  - El reporte original de Levin ya no se puede obtener en la base de datos de Lincoln Lab del MIT.
    - Pero sí se pueden obtener de 1965 en adelante.
    - ¿Coincidencia?

# Minimum Variance Distortionless Response

- Intenta minimizar la energía de las interferencias, por medio de:
  - Minimizar la energía de toda la salida
  - Excepto la que viene de la dirección del SOI
- Esta minimización la provee por medio del cálculo de un steering vector  $W_o$ , basado en un direction vector  $A$ , tal que:

$$\hat{S}_f = W_{o,f}^H X_f$$

# Minimum Variance Distortionless Response

- Por esta minimización, este filtro también es conocido como Minimum Power Distortionless Response.
  - Esta variación asume que las señales tienen poca correlación, por lo que MVDR es más general.

# Minimum Variance Distortionless Response

- Si tenemos:  $\hat{S}_{mvdr} = \mathbf{W}^H \mathbf{X}$
- La energía de la salida de es:

$$E_{salida} = |\hat{S}_{mvdr}|^2 = |\mathbf{W}^H \mathbf{X}|^2 = (\mathbf{W}^H \mathbf{X})(\mathbf{X}^H \mathbf{W}) = \mathbf{W}^H \mathbf{R} \mathbf{W}$$

Donde:

R: es la matriz de covariancia de las señales capturadas

W: es el steering vector óptimo a calcular

# Minimum Variance Distortionless Response

- Entonces la minimización se convierte en:

$$W_o = \arg \min_W (W^H R W)$$

- Con la siguiente restricción:

$$A^H W_o = 1$$

- De esta manera,  $W_o$  mantiene lo que proviene de la dirección de la SOI con relativamente alta energía.

# Minimum Variance Distortionless Response

- Cómo llevar a cabo esa minimización es algo complicado.
- Pero se encontró una solución general:

$$W_o = \frac{R^{-1} A}{A^H R^{-1} A}$$

# Minimum Variance Distortionless Response

- Cálculo de R:
  - Recordemos que A, así como lo fue W con D.A.S., se calcula por cada frecuencia.
  - De igual manera, se debe calcular una R por cada frecuencia (igual que MUSIC), utilizando la información de dicha frecuencia de todas las señales a lo largo del tiempo:

$$R_f = X_{f,1:T} X_{f,1:T}^H$$



# Minimum Variance Distortionless Response

- Inversión de  $R$ :
  - Desgraciadamente, es muy posible que la matriz de covariancia no sea invertible.
  - Para forzar su inversión, se recomienda multiplicar los valores en su diagonal por un valor justo arriba de 1. Por ejemplo, con una  $R$  de tamaño 2:

$$R \leftarrow R \cdot \begin{bmatrix} 1.001 & 1 \\ 1 & 1.001 \end{bmatrix}$$

# Ejercicio #2

- Descargar:
  - mvdr.m
- Prepara y simula las señales, y muestra las mismas figuras que el script das.
- Tiene la opción de amplificar la salida por medio de la variable:  
amp\_out

# Ejercicio #2

- Probar primero con los valores que vienen.
- Luego probar con:

$$M = 2$$

# Ejercicio #2

- Probar primero con los valores que vienen.
- Luego probar con:  
 $M = 2$
- Funciona muy bien con pocos micrófonos.

# Ejercicio #2

- Probar con:

`doa_steer = doa2`

`M = 2`

# Ejercicio #2

- Probar con:

`doa_steer = doa2`

`M = 2`

- Sigue funcionando bien.
- Tiene una pequeña tendencia de la primera señal.

# Minimum Variance Distortionless Response

- Beneficios:
  - Reducir la cantidad de micrófonos no lo impacta tanto como D.A.S.
- Problemas:
  - La inversión de la matriz de covariancia, por cada frecuencia, puede ser lento para llevarlo a cabo en línea.

# Minimum Variance Distortionless Response

- **ADVERTENCIA:**
- El principal objetivo de MVDR es minimizar la energía en la salida.
- Es muy posible que se requiere adivinar (en una sólo ocasión) cuánto se tiene que amplificar.
- En estas pruebas no fue necesario, porque son casos óptimos.
  - Sin ruido, sin reverberación, etc.



# Minimum Variance Distortionless Response

- **Variación “en línea”:**

- Otra forma de calcular R es recursivamente:

$$R_f[t+1] = \alpha R_f[t] + (1 - \alpha) X_{f,t} X_{f,t}^H$$

- Donde:

- $\alpha$  es una constante de suavamiento:

- $0 < \alpha < 1$ , pero cercano a 1.

- $X_{f,t}$  es el vector de valores de la entrada en la frecuencia  $f$  en el tiempo  $t$ .

- $X_{f,t} X_{f,t}^H$  es la *covariancia instantánea* (como discutido en MUSIC).

- Se deja como trabajo propio implementar y probar esta variación.

# Linearly Constrained Minimum Variance (LCMV)

# Linearly Constrained Minimum Variance

- Es una evolución de MVDR, en el que:
  - Se pretende minimizar la energía.
  - Manteniendo la dirección de SOI intacta.
  - Y, cancelando las direcciones de interferencias conocidas.

# Linearly Constrained Minimum Variance

- Usando el mismo modelo que MVDR:

$$W_o = \arg \min_W (W^H R W)$$

- Pero ahora con la siguiente restricción:

$$C^H W_o = [1 \ 0 \ 0 \ \cdots \ 0]$$

$$\text{Donde: } C = [A_{soi} \ A_{interf_1} \ A_{interf_2} \ \cdots]$$

$A_{soi}$ : es el direction vector de la SOI

$A_{interfi}$ : es el direction vector de la interferencia  $i$

# Linearly Constrained Minimum Variance

- Si aplicamos la solución general que se encuentra con MVDR, se obtiene:

$$W_{arr} = \frac{R^{-1}C}{C^H R^{-1}C}$$

- El resultado de la minimización entrega en  $W_{arr}$  un renglón por cada steering vector en  $C$ .
- El que nos interesa es el primero:

$$W_o = W_{arr}(:, 1)$$

# Ejercicio #3

- Descargar:
  - lcmv.m
- Igual al script mvdr.

# Ejercicio #3

- Probar primero con los valores que vienen.
- Luego probar con:

$$M = 2$$

# Ejercicio #3

- Probar primero con los valores que vienen.
- Luego probar con:  
 $M = 2$
- Funciona igual de bien con pocos micrófonos.



# Ejercicio #3

- Con los mismos valores, pero ahora:  
    `doa_steer = doa2`  
    `doa_null = doa1`

# Ejercicio #3

- Con los mismos valores, pero ahora:

`doa_steer = doa2`

`doa_null = doa1`

- Bastante bien.
- No existe la tendencia pequeña de la otra señal como sucedió con MVDR.

# Linearly Constrained Minimum Variance

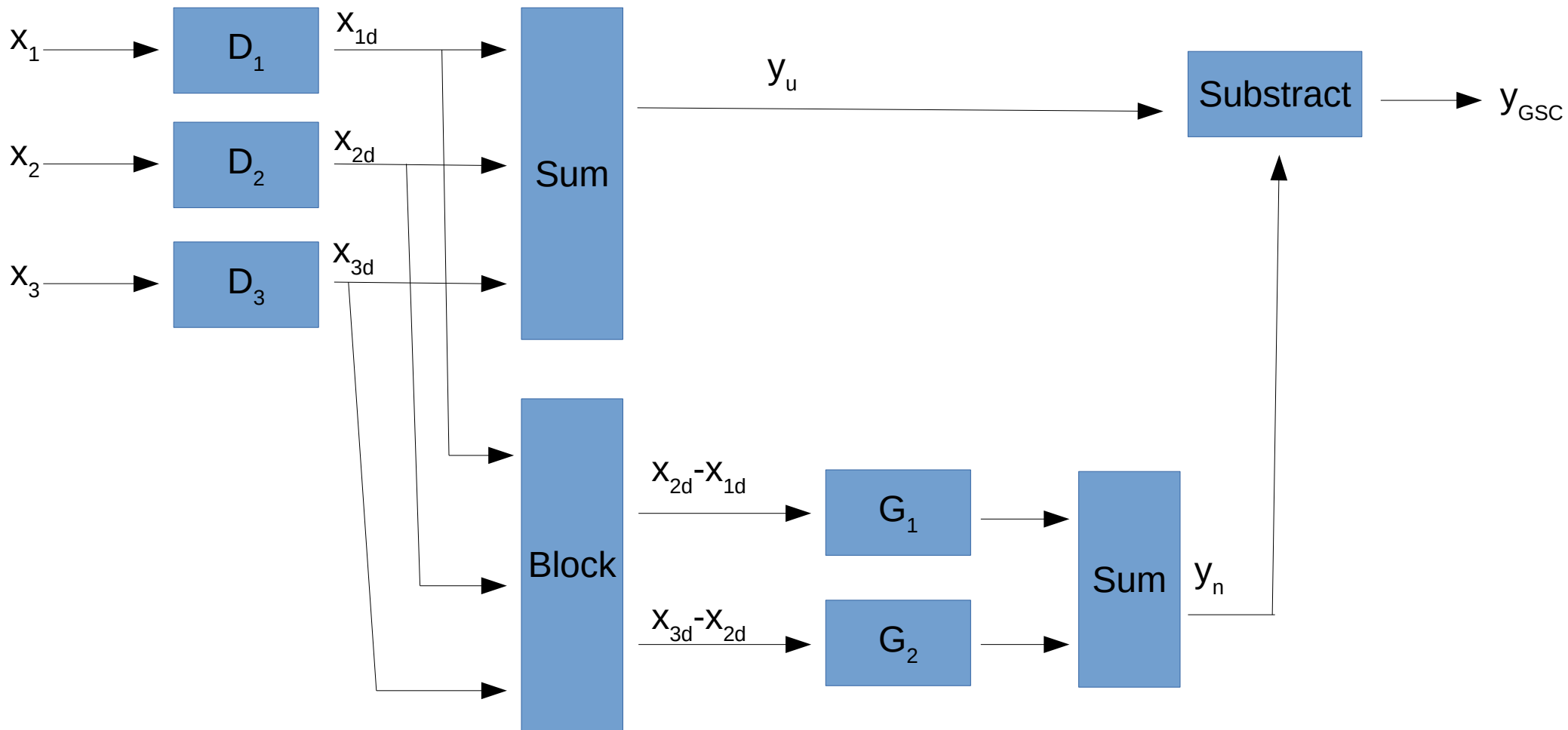
- Beneficios:
  - Puede funcionar con pocos micrófonos, como MVDR.
  - Más robusto a interferencias que MVDR.
- Problemas:
  - Mismos problemas que MVDR.
    - Inversión de R
    - Recalibración de `amp_out` al cambiar de SOI
      - En ambientes reales.
  - **Se requiere conocer la dirección de las interferencias.**

# Generalized Sidelobe Canceller (GSC)

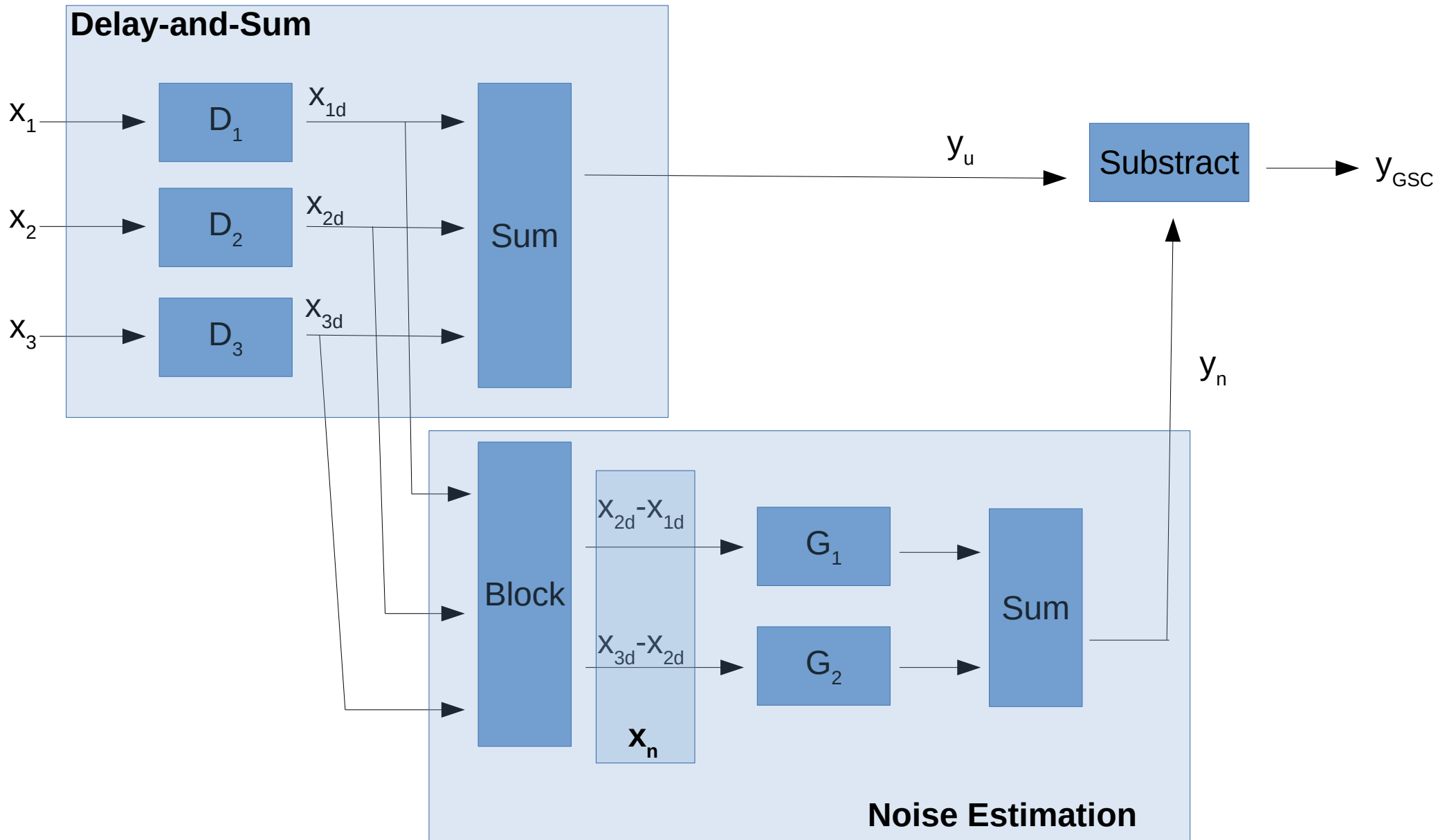
# Generalized Sidelobe Canceller

- Es una generalización de LCMV.
- En vez de cancelar direcciones específicas de las interferencias conocidas.
- Se cancela TODO lo que no venga de la dirección del SOI.

# Generalized Sidelobe Canceller



# Generalized Sidelobe Canceller



# Generalized Sidelobe Canceller

- $y_u$  : Salida del D.A.S.
  - Va a tener bastante información de la SOI, con algo de las interferencias.
- $y_n$  : Estimación de ruido.
  - Va a tener bastante información de las interferencias, con algo de la SOI.
- $y_{GSC}$  : Salida del sistema.

$$Y_{GSC} = y_u - y_n$$



# Matriz de Bloqueo

- El bloque que dice “Block” es conocido como la *matriz de bloqueo*.
  - En este caso, la matriz de bloqueo es equivalente a restar las señales adyacentes.
  - Hay otras formatos de esta matriz, pero esta versión es la más simple que, a su vez, es efectiva.

# Generalized Sidelobe Cancellor

- Objetivo:
  - Encontrar una serie de filtros  $G_x$  que, al aplicarlos a  $y_n$ , la resta provee una salida sin casi nada de interferencias.
- Desgraciadamente, éste es el problema principal de esta técnica:
  - No hay serie de filtros  $G_x$  que sean aplicables a todas las combinaciones de señales que existen.

# Adaptabilidad

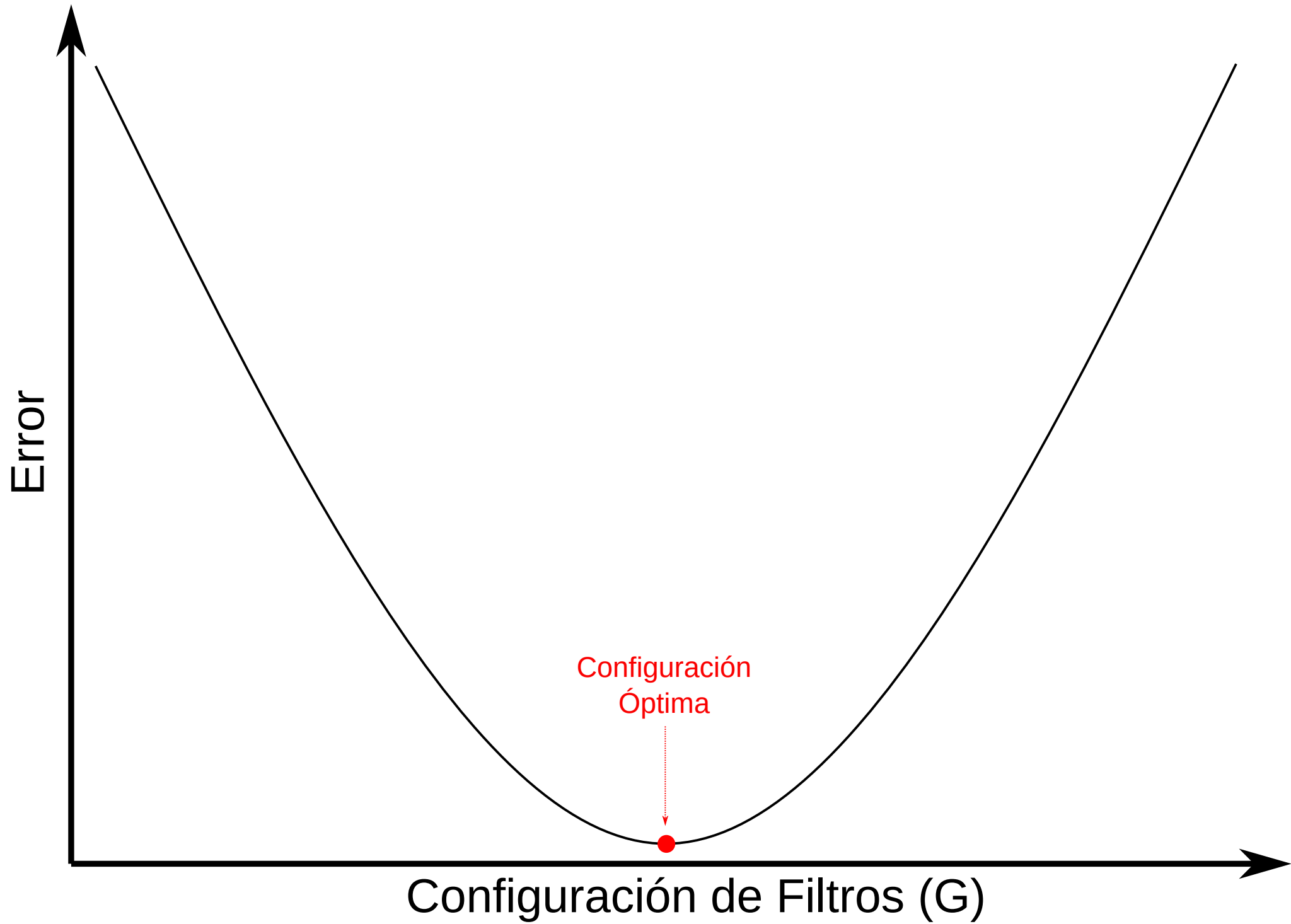
- Pero, la forma del problema da a lugar a que se pueda aplicar un sistema de adaptación para que el propio sistema encuentre los  $G_x$  que sean los más apropiados a la SOI y ruidos estimados.

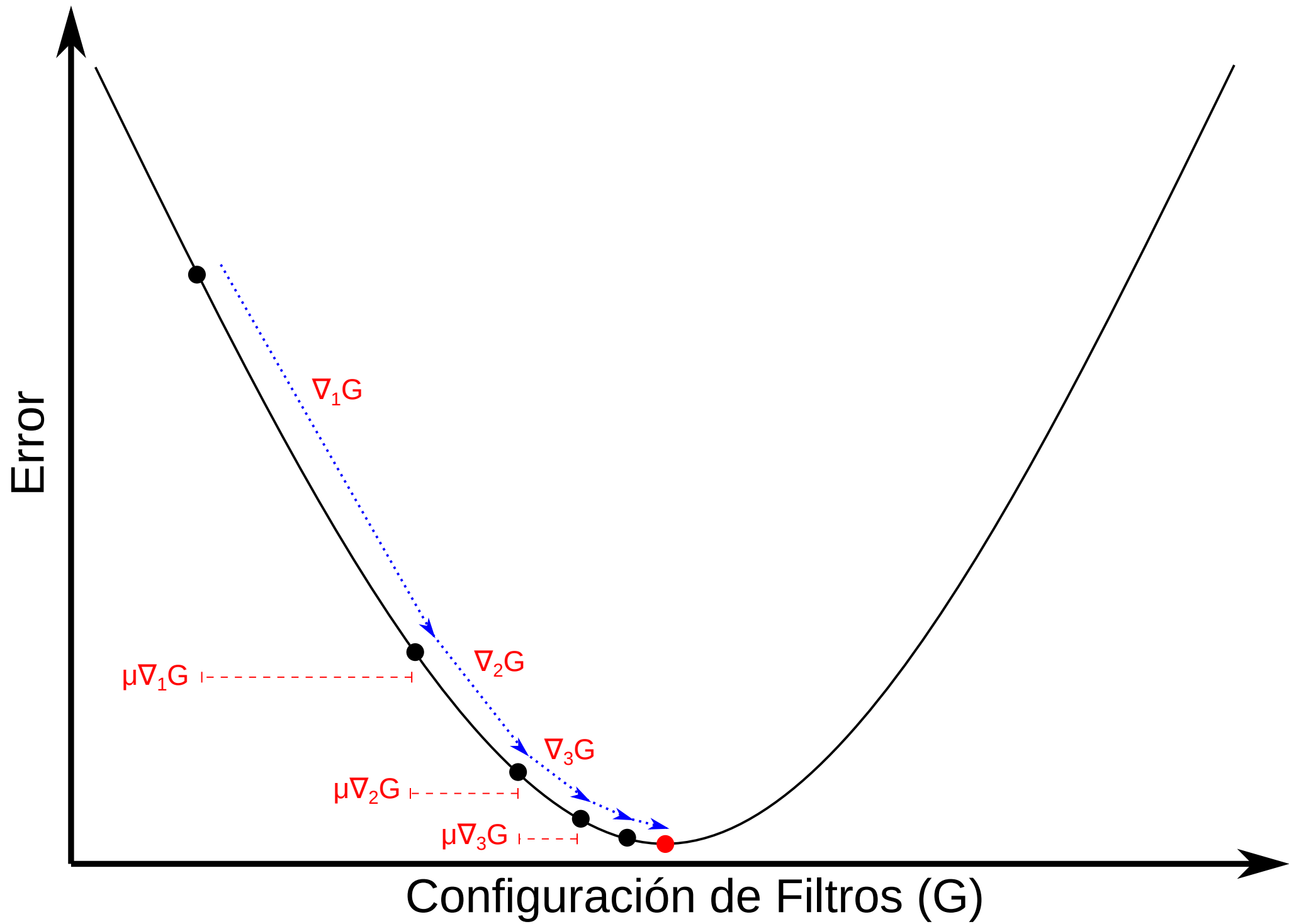
# Algoritmo de Mínimos Cuadrados

- Es una forma de encontrar un filtro óptimo (dada una señal e interferencias), en línea.
- Se encuentra por medio de minimizar el error entre la señal deseada y la señal estimada.

# Algoritmo de Mínimos Cuadrados

- Esta minimización se busca a partir de:
  - Calcular el gradiente más inclinado entre la versión de los filtros pasados y los actuales.
    - Esta gradiente se calcula en base al error entre la señal deseada y la estimada.
  - Utilizar el gradiente (multiplicado por una constante predefinida) para calcular un nuevo conjunto de filtros.
  - Utilizar dichos filtros para estimar una nueva señal.





# Algoritmo de Mínimos Cuadrados

- La gradiente más inclinada se calcula así:

$$\nabla G = y e$$

- Por lo tanto, la actualización de los filtros se hace así:

$$\mathbf{G}_{k+1} = \mathbf{G}_k + \mu y e$$

Donde:

$\mu$ : es la constante de adaptación.

Entre más alta, más grandes serán los pasos de adaptación.

La multiplicación es punto-a-punto



# Algoritmo de Mínimos Cuadrados

- Para nuestros propósitos:

y: es la salida del sistema

e: es el ruido que se ha estimado ( $x_n$ )

$$\mathbf{G}_{k+1} = \mathbf{G}_k + \mu y_{GSC} \mathbf{x}_n$$

Donde:

$\mu$ : es la constante de adaptación.

Entre más alta, más grandes serán los pasos de adaptación.

La multiplicación es punto-a-punto, para cada renglón de  $x_n$

# Generalized Sidelobe Canceller

- Esto implica que el sistema no puede calcular toda la señal a la vez.
- Tiene que hacerlo a lo largo del tiempo de muestreo.
  - Lo cual se acomoda muy bien a nuestras necesidades.

# Generalized Sidelobe Cancellor

- Dado una dirección del SOI
  - Desfasar las señales de entrada apropiadamente.
- Por cada muestra en un momento  $k$ :
  - Obtener de las señales desfasadas las muestras  $k-N$  hasta  $k$ .
    - $N$  es el tamaño del filtro.
  - Calcular la salida del D.A.S. por medio de sumar punto a punto las señales desfasadas:  $y_u$
  - Calcular el resultado de la matriz de bloqueo, por medio de restar las señales adyacentes, y guardar en buffer:  $x_n$
  - Aplicar los filtros  $G$  al buffer de los resultados de la matriz de bloqueo, para obtener una estimación del ruido:  $y_n = Gx_n$
  - Obtener la salida del GSC:  $y_{GSC} = y_u - y_n$
  - Actualizar los filtros con la información calculada:  $G = G + \mu y_{GSC} x_n$

# Ejercicio #4

- Descargar:
  - gsc.m
- Igual al script mvdr, sólo que ahora también esta al variable de:
  - mu: la cual es equivalente a  $\mu$

# Ejercicio #4

- Probar primero con los valores que vienen.

# Ejercicio #4

- Probar primero con los valores que vienen.
- Hay momentos al principio de la señal ruidosos.
  - Pero después ya funciona moderadamente bien.

# Ejercicio #4

- Probar con:  
     $\text{doa\_steer} = \text{doa2}$   
     $\mu = 0.05$

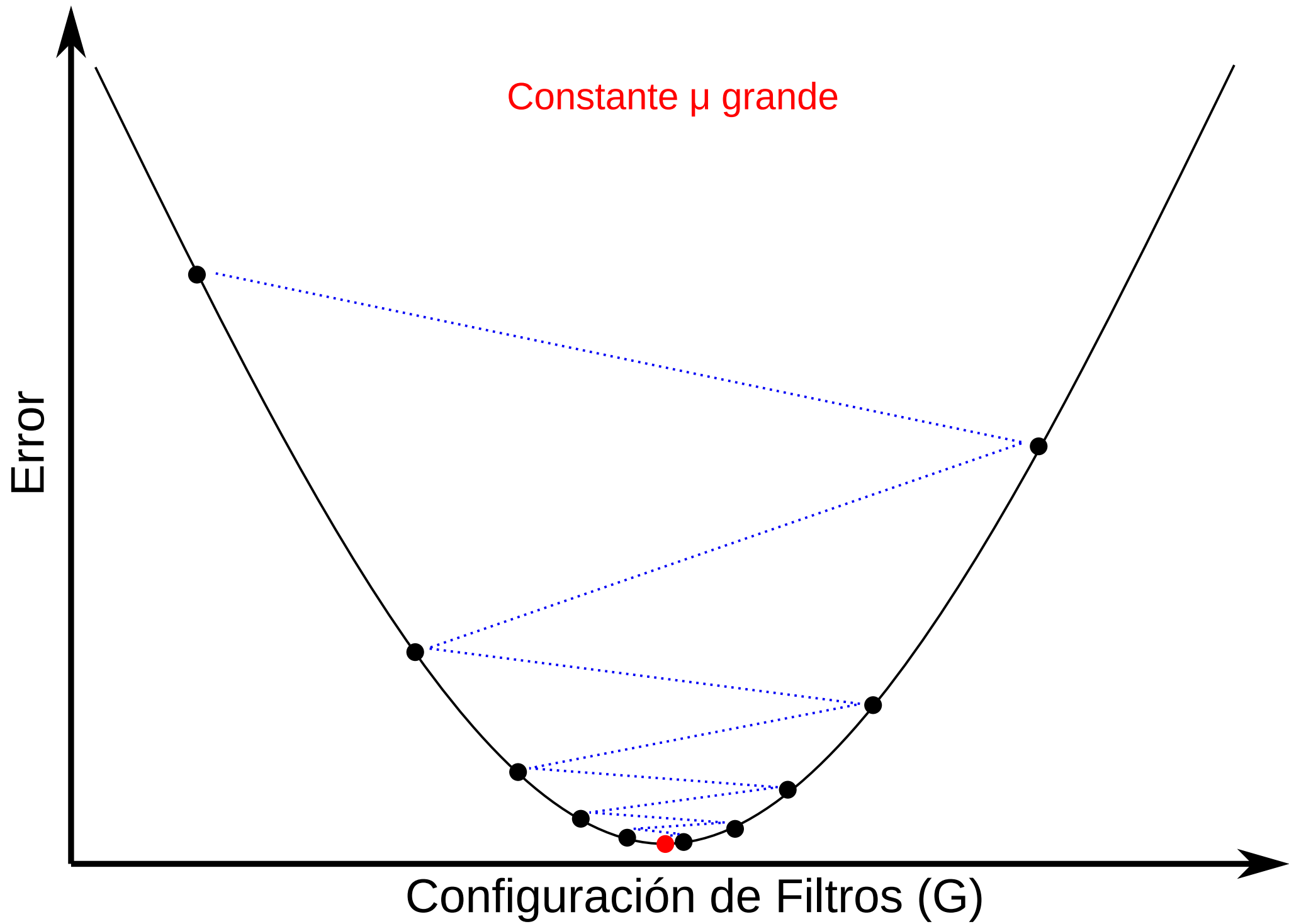
# Ejercicio #4

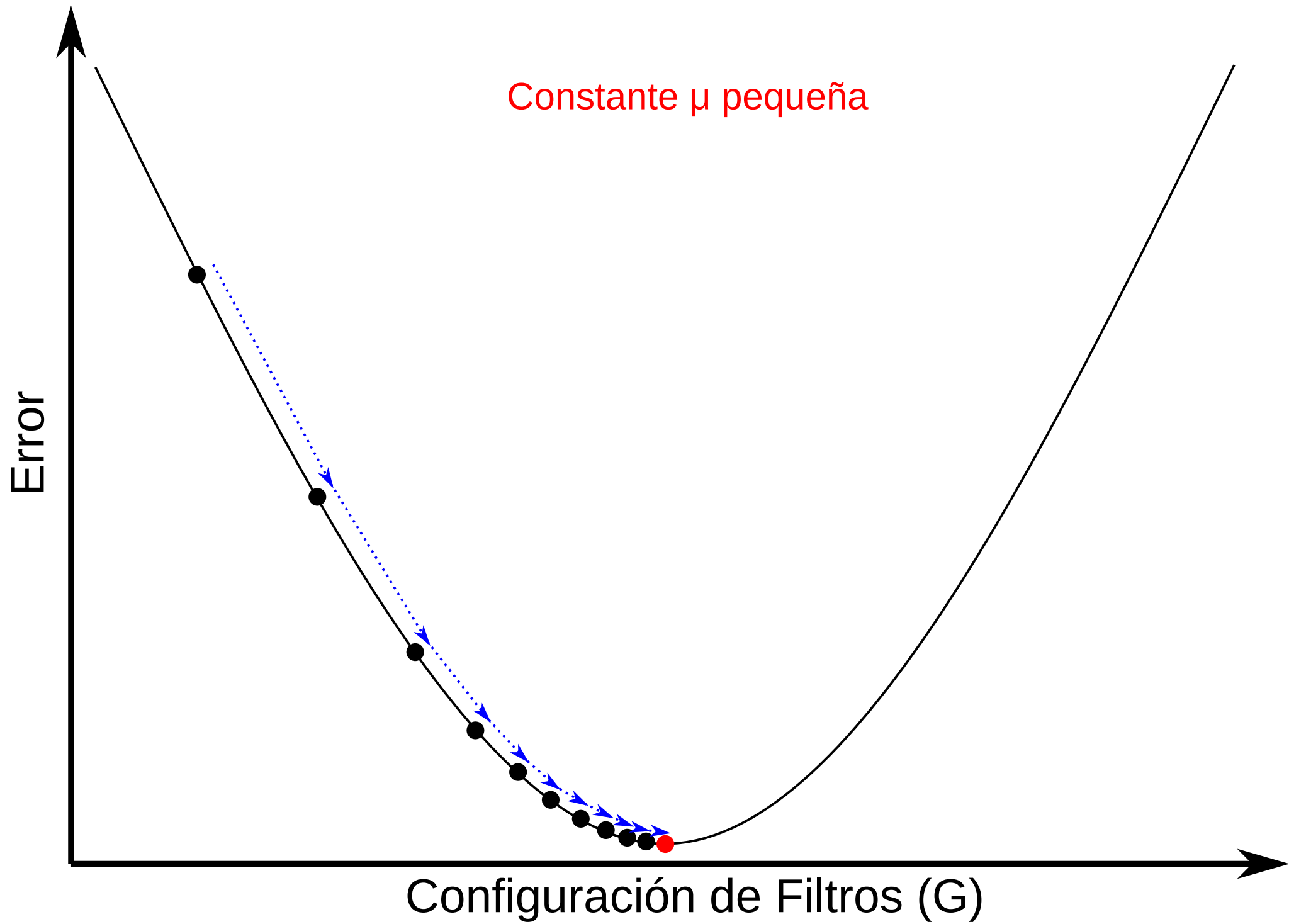
- Probar con:  
     $\text{doa\_steer} = \text{doa2}$   
     $\mu = 0.05$
- Se deteriora la calidad.
  - Es sensible al SIR.
- Al cambiar la señal, también se tiene que cambiar  $\mu$ .



# Ejercicio #4

- Se puede observar que en algunos momentos la señal está bien, pero en otros no.
- Para esto se propone mejor utilizar una  $\mu$  dinámica, que se modifique dependiendo del SIR presente.





# GSC con $\mu$ Dinámico

Para cada  $x_n$ , hay un filtro  $g_n$  que se actualiza con su propio  $\mu_n$ , calculado a partir de la razón entre las energías de la salida y el ruido  $x_n$ :

$\mu_0$  :  $\mu$  inicial

$\mu_{\max}$  :  $\mu$  máximo

$p_{x_n}$  : energía de  $x_n$

$p_y$  : energía de  $y_{\text{GSC}}$

```
if ( $\mu_0 * p_{x_n} / p_y < \mu_{\max}$ )
```

```
     $\mu = \mu_0 / p_y$ ;
```

```
else
```

```
     $\mu = \mu_0 / p_{x_n}$ ;
```

```
end
```

Un ejemplo completo se puede observar en `gsc_dyn`

# Ejercicio #5

- Descargar:
  - gsc\_dyn.m
- Igual al script gsc, sólo que en vez de la variable mu, ahora también están:
  - mu0: es una  $\mu$  “inicial”
  - mu\_max: es el límite máximo de  $\mu$

# Ejercicio #5

- Probar primero con los valores que vienen.

# Ejercicio #5

- Probar primero con los valores que vienen.
- Desempeño parecido a gsc.
  - Aunque sigue un poco mejor a la señal de interés.

# Ejercicio #5

- Ahora probar con:  
    `doa_steer = doa2`



# Ejercicio #5

- Ahora probar con:  
`doa_steer = doa2`
- Batalla en adaptarse al inicio, pero se acomoda a la señal de interés al final.
- No requirió recalibrar a  $\mu_0$  ni  $\mu_{\max}$ .

# Ejercicio #5

- Ahora probar con:  
    `doa_steer = doa1`  
     $M = 3$

# Ejercicio #5

- Ahora probar con:  
doa\_steer = doa1  
M = 3
- Perdió calidad en la señal de salida.

# GSC con $\mu$ Dinámico

- Beneficios:
  - Compatible a soluciones en línea.
  - Muy liviano.
  - No requiere ir a y regresar del dominio de la frecuencia.
  - Buenos resultados, dado un tiempo de adaptación.

# GSC con $\mu$ Dinámico

- Problemas:
  - Requiere de tiempo para encontrar los filtros óptimos durante el muestreo.
  - Requiere de calibración de la constante de adaptación para cada diferente ambiente sonoro.
    - Dos variables a calibrar ( $\mu_0$  y  $\mu_{\max}$ ).
  - Requiere de una cantidad moderada (8) de micrófonos.

# Frequency Masking

# Frequency Masking

- Realmente no es Beamforming.
  - De hecho, requiere un sólo micrófono.
- Pero lo agregó aquí porque con algunos ajustes, se parece mucho a una técnica de Beamforming.

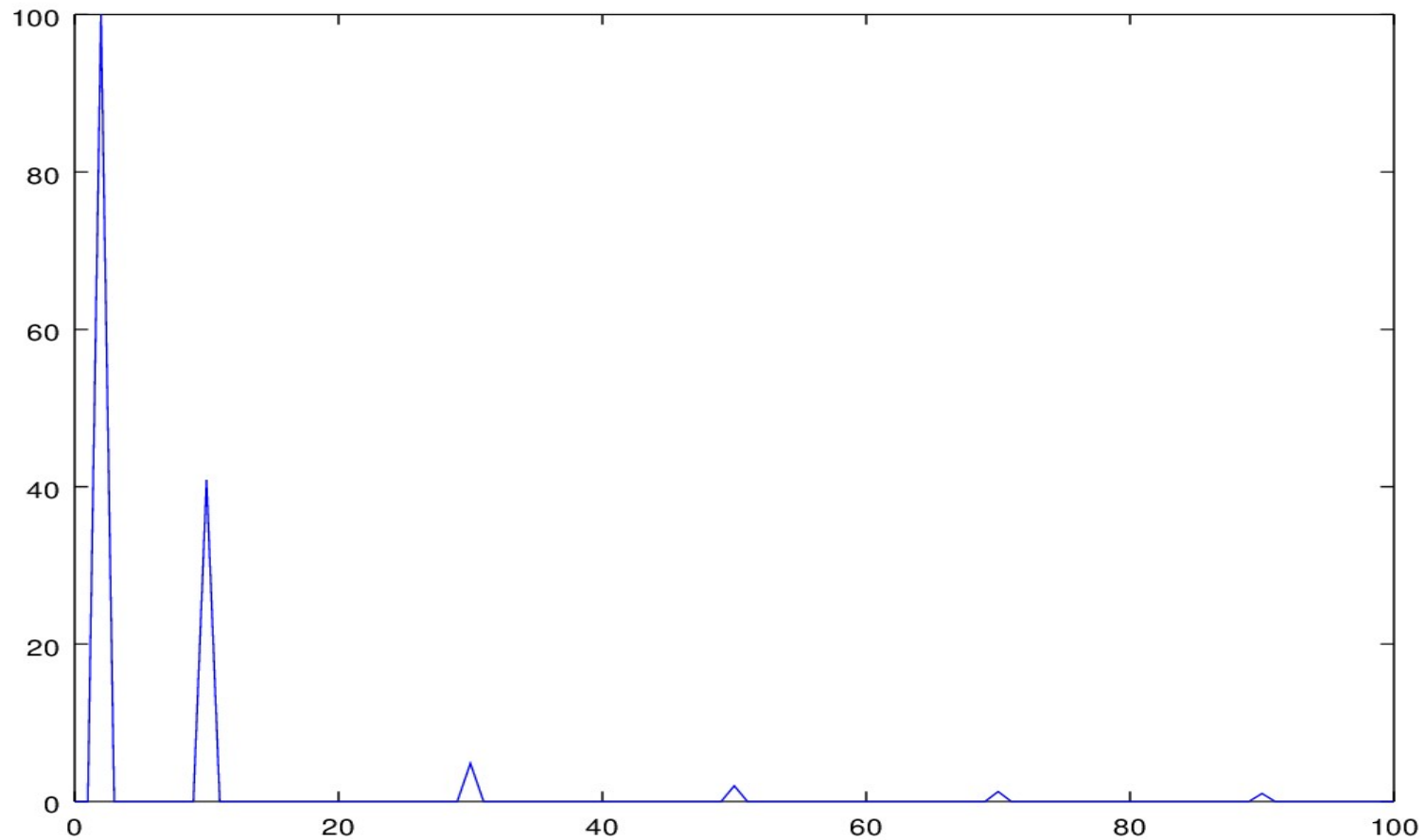
# Frequency Masking

- La intención es encontrar una máscara óptima tal que al aplicarla a nuestra entrada en el dominio de la frecuencia, sólo permita pasar las frecuencias de nuestra señal de interés.



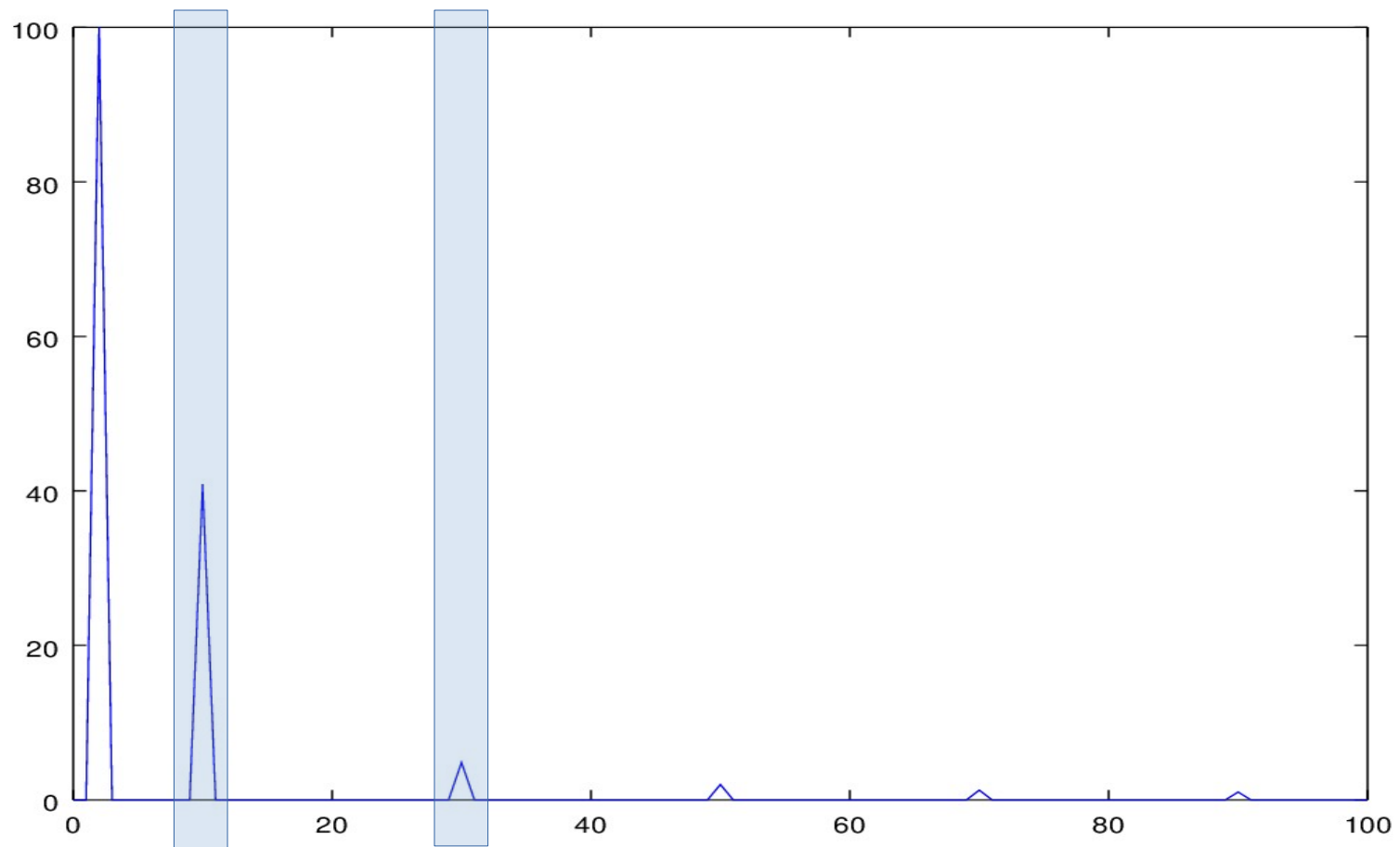
# Frequency Masking

- Es decir, dado un espectro como:



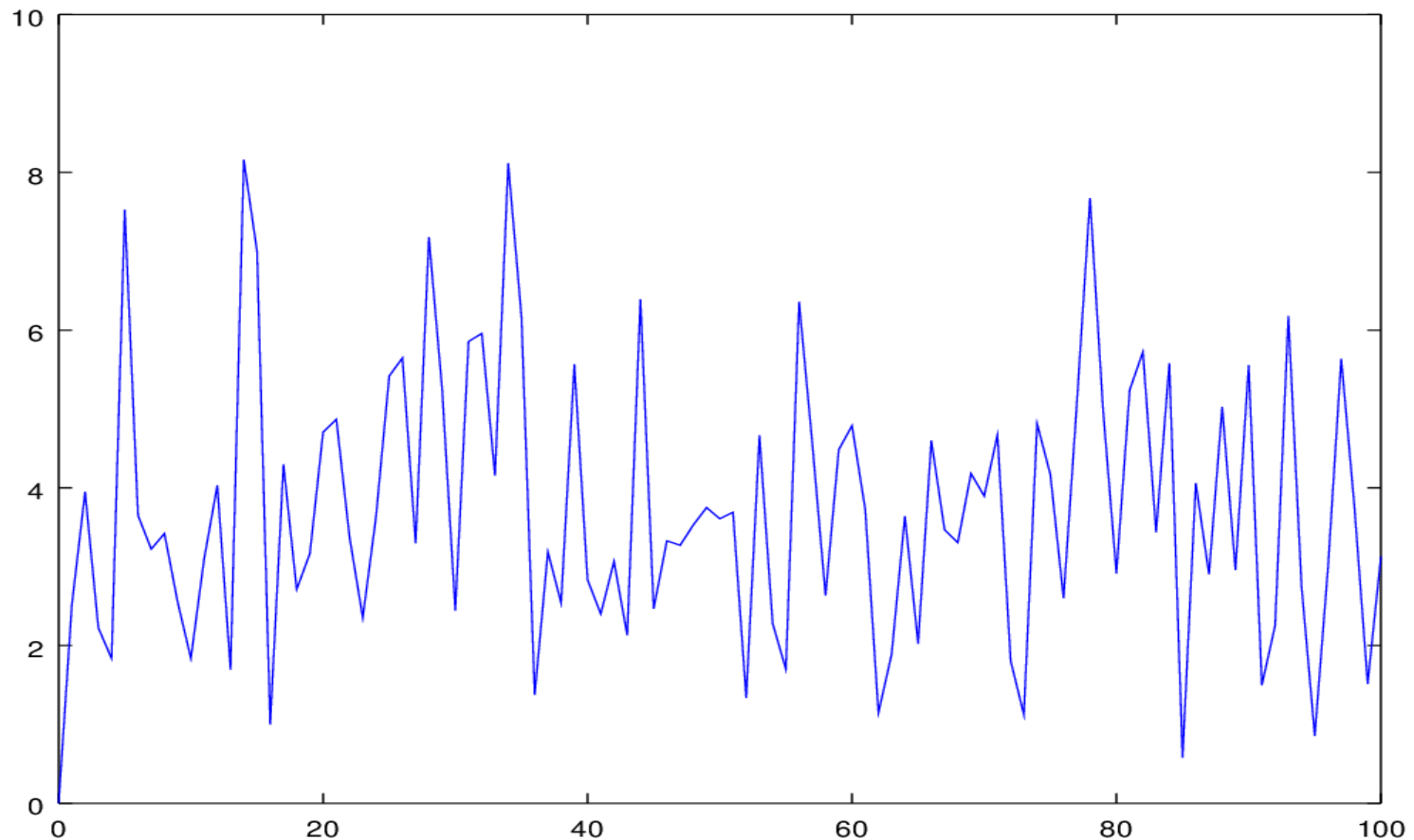
# Frequency Masking

- La máscara selecciona frecuencias como:



# Frequency Masking

- Pero cuando tenemos espectros como:



# Si los unicornios fueran reales...

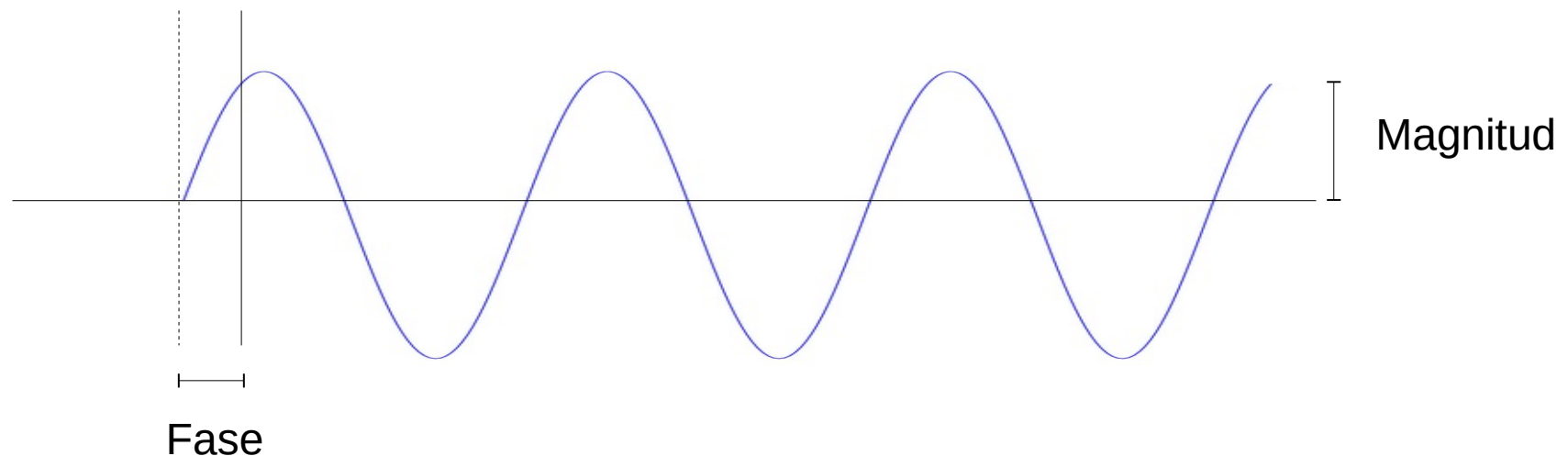
- Esta dichosa máscara óptima es el santo grial para muchos de la comunidad de separación de fuentes.
- Desafortunadamente, entre más y más se busca, más se cree que no existe.
- Y si existe, es dinámica, ya que depende de:
  - Presencia/características de las interferencias.
  - El entorno acústico.
  - Cambios en la misma señal de interés.

# ¿Entonces?

- Se puede diseñar dicha máscara utilizando información de beamforming.

# Recordatorio

- La fase de una frecuencia es el retraso en radianes de la señal senoidal que posee dicha frecuencia.



# Operador de Desfase

- Al multiplicar a una frecuencia “f” por el operador de desfase:

$$e^{-i2\pi f T}$$

- Se afecta directamente a la fase de dicha frecuencia.

# Por lo tanto...

- Al estar llevando a cabo un Delay-and-Sum en el dominio de la frecuencia, lo que realmente está sucediendo:
  - Se “alinean” en fase las frecuencias pertenecientes a nuestra SOI.
  - Y se “desalinean” aquellas que no.
- De esta manera, al sumarse, se intensifican las frecuencias de nuestro SOI, y se reducen las que no.
- Esto significa que las frecuencias que tengan una fase similar en todas las señales, son frecuencias que pertenecen a nuestra señal de interés.



# Phase-Based Frequency Masking

- Aplicar  $W^H$  a  $X$ , pero sin sumar las señales.
  - Multiplicación punto-a-punto, no matricial.
- Ahora, por cada frecuencia:
  - Calcular la diferencia de la fase entre las señales resultantes.
  - Si la diferencia es menor a un umbral, la frecuencia pertenece a nuestro SOI.
    - Escribir en la salida el valor de esta frecuencia del micrófono de referencia.
  - Si no, no pertenece.
    - Escribir en la salida un 0 en esta frecuencia.

# Ejercicio #6

- Descargar:
  - freqmask.m
- Crea las siguientes figuras:
  - Figura 1: las señales de origen.
    - Tiempo arriba; magnitud en frecuencia abajo.
  - Figure 2: la señal capturada en el primer micrófono.
    - Tiempo arriba; magnitud en frecuencia en medio; fase en frecuencia abajo.
  - Figure 3: señales de origen superpuestas sobre el resultado.
    - Tiempo arriba; magnitud en frecuencia en medio; fase en frecuencia abajo.

# Ejercicio #6

- Probar primero con los valores que vienen.

# Ejercicio #6

- Probar primero con los valores que vienen.
- Funciona bien.
  - La máscara (en rojo) selecciona sólo el primer pico.
    - La primera señal es senoidal, sólo ocupa un pico.

# Ejercicio #6

- Probar ahora con:  
    `doa_steer = doa2`

# Ejercicio #6

- Probar ahora con:  
`doa_steer = doa2`
- Sigue funcionando bien.
  - La máscara ahora agarra al resto de los picos.

# Phase-Based Frequency Masking

- El tipo de máscara que se está utilizando se conoce como **binaria**.
- Este tipo de máscaras introducen discontinuidades en el dominio de la frecuencia.
  - Es típico escuchar artefactos en la salida.

# Phase-Based Frequency Masking

- Beneficios:
  - Casi tan liviana como DAS.
- Problemas:
  - Discontinuidades en frecuencia y tiempo.



# Antes de terminar con este tema...

- Es importante hacer notar que las pruebas que se hicieron aquí son representativas, pero no exhaustivas.
- El código que se entrega es para que ustedes hagan sus propias pruebas
- Así, obtengan sus propias conclusiones de cuál técnica es la que utilizarán en su proyecto final.

# Ideas a probar

- ¿Cuál es el impacto de la distancia entre micrófonos para cada técnica?
- Si se cambia el número de micrófonos, la distancia máxima entre el micrófono de referencia y el último micrófono también cambia. ¿Cómo impacta esto a las técnicas?
- ¿Qué pasa si se incrementa la intensidad de la señal triangular a que sea más grande que la senoidal?
- ¿Qué pasa si se utilizan otros tipos de señales?

Siguiente Clase:

Separación de Fuentes por Análisis Estadístico