Separación de Fuentes:

Beamforming

Modelo de Señales Capturadas

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

$$\boldsymbol{A}=\begin{bmatrix} e^{-i2\pi fT_{1:1}} & e^{-i2\pi fT_{1:2}} & \cdots & e^{-i2\pi fT_{1:M}} \\ e^{-i2\pi fT_{2:1}} & e^{-i2\pi fT_{2:2}} & \cdots & e^{-i2\pi fT_{2:M}} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-i2\pi fT_{D:1}} & e^{-i2\pi fT_{D:2}} & \cdots & e^{-i2\pi fT_{D:M}} \end{bmatrix} \qquad \boldsymbol{S}=\begin{bmatrix} s_1(1) & s_1(2) & \cdots & s_1(N) \\ s_2(1) & s_2(2) & \cdots & s_2(N) \\ \vdots & \vdots & \ddots & \vdots \\ s_M(1) & s_M(2) & \cdots & s_M(N) \end{bmatrix}$$

$$\boldsymbol{X}=\boldsymbol{S}\,\boldsymbol{A}$$

Donde:

s_x: es una señal de origen

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

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

A: es la matriz que contiene los vectores de dirección (*direction vectors*)

X: son las señales capturadas (en los micrófonos); cada renglón representa un micrófono

Objetivo

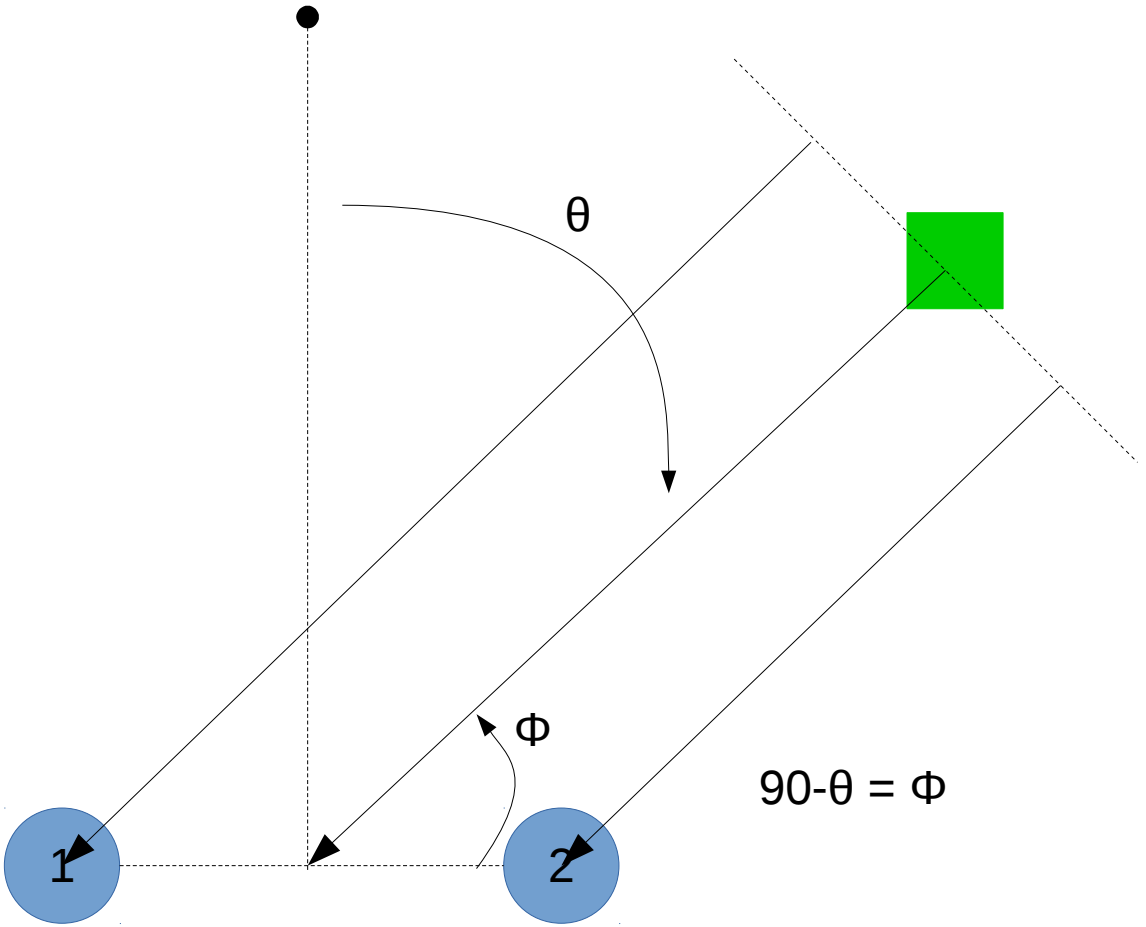
- Estimar las señales en **S** por medio de aplicar una matriz adicional **W** a **X**.

$$\hat{\boldsymbol{S}} = \boldsymbol{W}\,\boldsymbol{X}$$

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



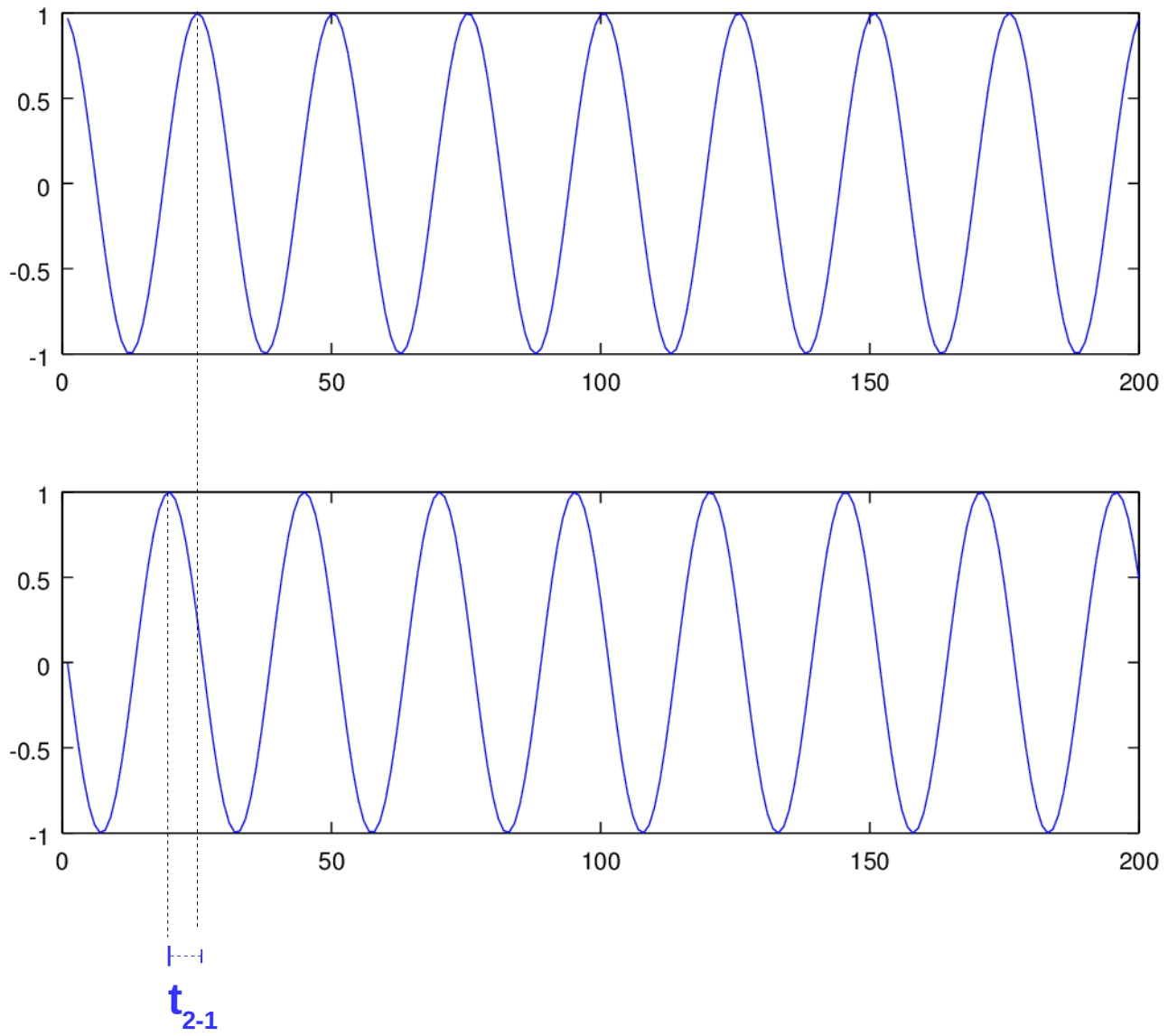
Beamforming

- 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

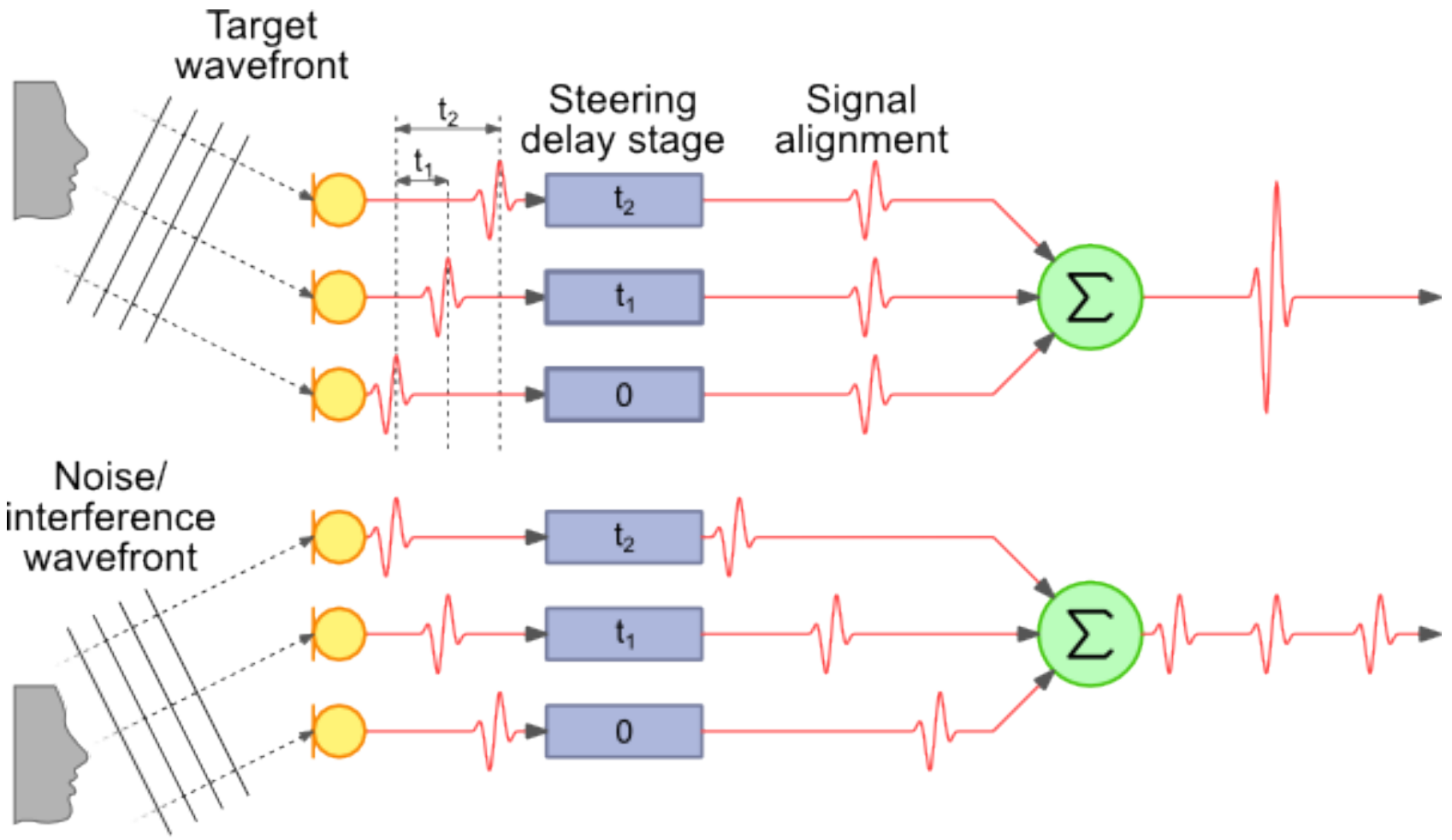
Señal 2



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 calculá el desfase apropriado 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.

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:

$$g(t-T) = F^{-1}\left(G(t)\,e^{-i\,2\pi fT}\right)$$

- 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. $e^{-i\,2\pi fT}$
  - Esto es el “steering vector” del arreglo: **W**.
- Hacerle FFT a las señales de entrada.
- Aplicar: $$\hat{\boldsymbol{S}} = \boldsymbol{W}^{\boldsymbol{H}}\,\boldsymbol{X}$$
- Sacarle la IFFT a S, el cual es la salida.

^{H}: es la operación de transposición conjugada de una matriz

¿Por que transpuesta conjugada?

W en Delay-and-Sum (DAS)

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

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

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

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: steering vector

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.

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

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).'*X(:,f);
- das entrega el mismo resultado.

Ejercicio #1

- ¿Qué sucede si reducimos el número de micrófonos?

Ejercicio #1

- ¿Qué sucede si reducimos el número de micrófonos?

- 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

- ¿Qué sucede si modificamos a doa_steer a apuntar a la otra señal de origen?

Ejercicio #1

- ¿Qué sucede si modificamos a doa_steer a apuntar a la otra señal de origen?

- La otra señal aparece en la salida.
  - Aunque no con buena calidad.
  - Es porque la señal de interferencia tiene más energía que la SOI.

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:

$$SIR = \frac{P_{SOI}}{P_{Interf}}$$

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.

Otras Formas de Beamforming

- Minimum Variance Distortionless Response
- Linearly Constrained Minimum Variance
- Generalized Sidelobe Canceller

Minimimum Variance Distiortionless Response (MVDR)

Minimimum Variance Distiortionless 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 A, basado en W, tal que:

$$\hat{S}_{mvdr} = A^{H} X$$

Minimimum Variance Distiortionless Response

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

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

Donde:

R: es la matriz de covariancia de las señales capturadas

A: es el steering vector óptimo a calcular, basado en el W de D.A.S Beamforming

Minimimum Variance Distiortionless Response

- Entonces las minimización se convierte en:

$$\boldsymbol{A}_{\boldsymbol{opt}} = arg\min_{A}\left(\boldsymbol{A}^{\boldsymbol{H}}\,\boldsymbol{R}\,\boldsymbol{A}\right)$$

- Con la siguiente restricción:

$$W^{H} A_{opt} = 1$$

- De esta manera, A mantiene con relativamente alta a energía a la dirección de la SOI.

Minimimum Variance Distiortionless Response

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

$$\boldsymbol{A}_{\boldsymbol{opt}} = \frac{R^{-1} W}{W^{H} R^{-1} W}$$

Minimimum Variance Distiortionless 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:

$$R(f) = X_{1:T}(f)\, X_{1:T}(f)^{H}$$

Minimimum Variance Distiortionless 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 = 3

  amp_out = 1.75

Ejercicio #2

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

  M = 3

  amp_out = 1.75

- Parece funcionar bien con pocos micrófonos.
  - Aunque hay unos problemas al inicio y final.

Ejercicio #2

- Probar con:

  doa_steer = doa2

  M = 8

  amp_out = 200

Ejercicio #2

- Probar con:

  doa_steer = doa2

  M = 8

  amp_out = 200

- Se tiene que amplificar la señal bastante.

Minimimum Variance Distiortionless Response

- Beneficios:
  - Reducir la cantidad de micrófonos no lo impacta tanto como D.A.S.
- Problemas:
  - Ya que su principal objetivo es minimizar la energía en la salida, es muy posible que se requiere adivinar (en una sóla ocasión) cuánto se tiene que amplificar.
  - La inversión de la matriz de covariancia, por cada frecuencia, puede ser lento para llevarlo a cabo en línea.

Linearly Constrained Minimium Variance

(LCMV)

Linearly Constrained Minimium 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 Minimium Variance

- Usando el mismo modelo que MVDR:

$$\boldsymbol{A}_{\boldsymbol{opt}} = arg\min_{A}\left(\boldsymbol{A}^{\boldsymbol{H}}\,\boldsymbol{R}\,\boldsymbol{A}\right)$$

- Pero ahora con la siguiente restricción:

$$C^{H} A_{opt} = \left[1\ 0\ \ 0 \cdots 0\right]$$

Donde: $C = \left[W\ N\right]$

N: es el steering vector en las direcciones de las interferencias conocidas

Linearly Constrained Minimium Variance

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

$$\boldsymbol{A}_{\boldsymbol{arr}} = \frac{R^{-1} C}{C^{H} R^{-1} C}$$

- El resultado de la minimización entrega en A un renglón por cada steering vector en C.
- El que nos interesa es el primero:

$$\boldsymbol{A}_{\boldsymbol{opt}} = A_{arr}(:,1)$$

Ejercicio #3

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

Ejercicio #3

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

  d = 0.5

  M = 5

  amp_out = 0.21

Ejercicio #3

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

  d = 0.5

  M = 5

  amp_out = 0.21

- Funciona fenomenal con distancias pequeñas entre micrófonos.

Ejercicio #3

- Con los mismos valores, pero ahora:

  doa_steer = doa2

  doa_null = doa1

  amp_out = 0.5

Ejercicio #3

- Con los mismos valores, pero ahora:

  doa_steer = doa2

  doa_null = doa1

  amp_out = 0.5

- Sigue funcionando, pero no tan bien.
  - No le gusta SIR bajos.
- Se tiene que recalibrar amp_out.

Ejercicio #3

- De nuevo probar con doa1 como SOI, pero ahora con:

  M = 3

  amp_out = 0.3

Ejercicio #3

- De nuevo probar con doa1 como SOI, pero ahora con:

  M = 3

  amp_out = 0.3

- Su desempeño se reduce un poco, pero no está tan mal.

Linearly Constrained Minimium Variance

- Beneficios:
  - Muy buen desempeño con distancias pequeños entre micrófonos.
  - Puede funcionar con pocos micrófonos.

Linearly Constrained Minimium Variance

- Problemas:
  - Mismos problemas de inversión de R que MVDR.
  - Necesita recalibración al cambiar de SOI o número de micrófonos.
    - Aunque no tanto como MVDR.
  - **Se requiere conocer la dirección de las interferencias.**

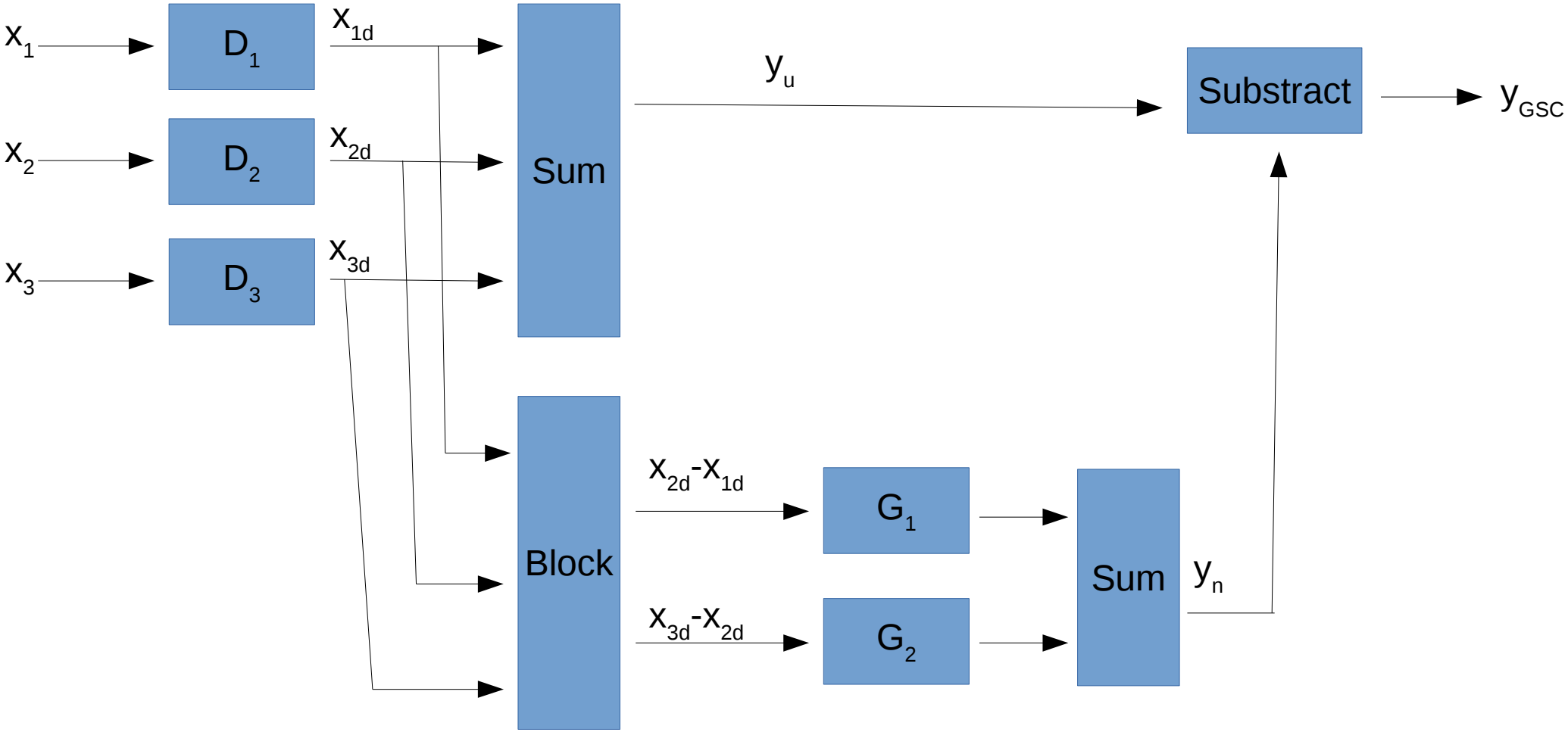
Generalized Sidelobe Canceller

(GSC)

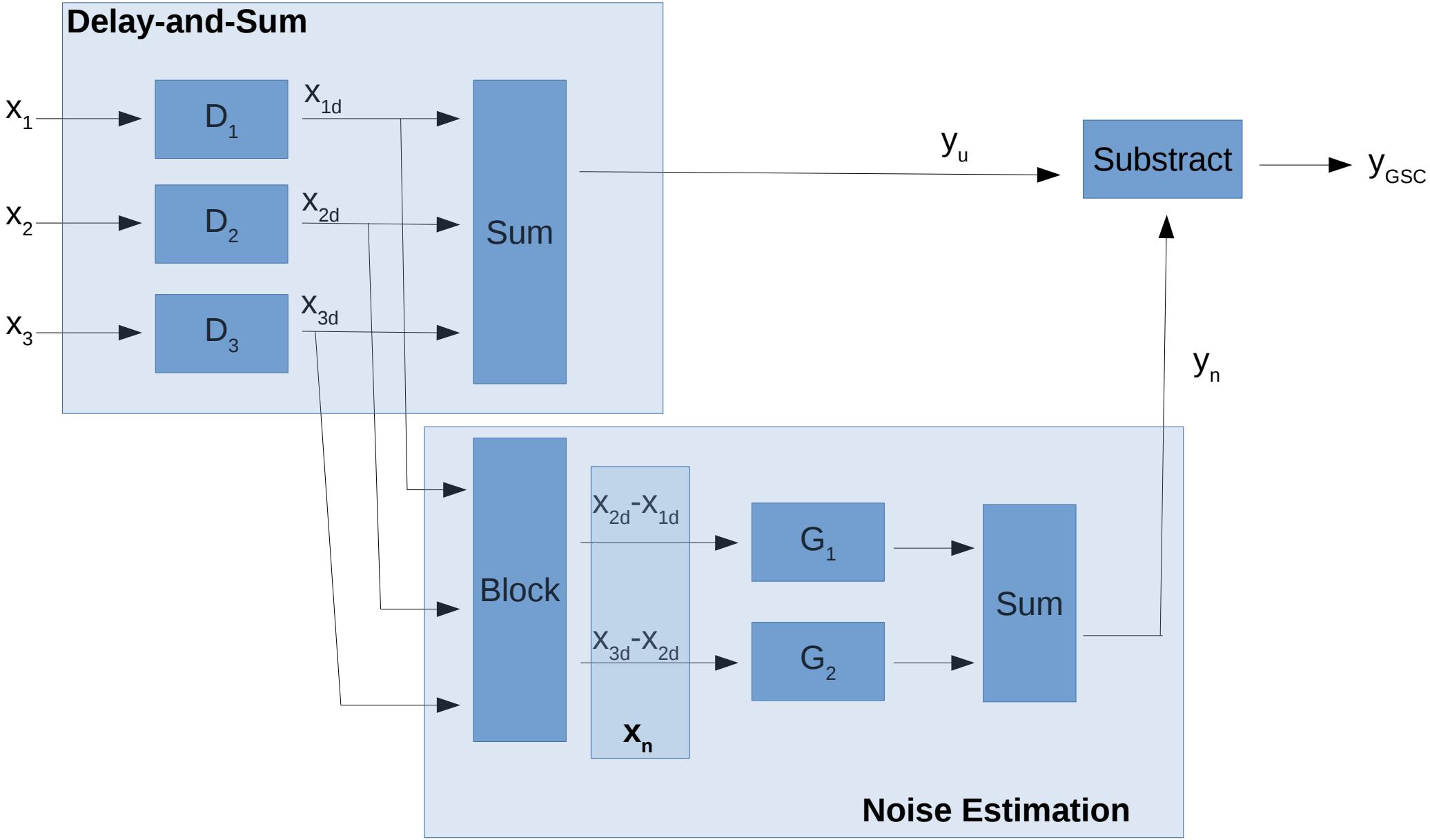
Generalized Sidelobe Canceller

- 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 Canceller

- Objetivo:
  - Encontrar una serie de filtros G_x que, al aplicarlos a y_n, la resta provee una salida sin casi nada de interferencias.
- Desgraciademente, 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 un peso predefinido) 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:

$$\boldsymbol{G_{k+1} = G_k + \mu\, y\, e}$$

Donde:

μ: 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)

$$G_{k+1} = G_k + \mu\, y_{GSC}\, x_n$$

Donde:

μ: 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 Canceller

- Dado una dirección del SOI
  - Desfasar las señales de entrada apropriadamente.
- 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: x_n
  - Aplicar los filtros G al resultado 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 μ

Ejercicio #4

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

  doa_steer = doa2

  amp_out = 0.85

  mu = 0.03

Ejercicio #4

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

  doa_steer = doa2

  amp_out = 0.85

  mu = 0.03

- En ambos casos, hay momentos al principio de la señal ruidosos.
  - Pero después ya funciona bien.
- Al cambiar la señal, también se tiene que cambiar a mu.

Ejercicio #4

- Ahora probar con:

  doa_steer = doa1

  amp_out = 0.95

  mu = 0.005

  M = 3

Ejercicio #4

- Ahora probar con:

  doa_steer = doa1

  amp_out = 0.95

  mu = 0.005

  M = 3

- Perdió calidad en la señal de salida.

Generalized Sidelobe Canceller

- 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.

Generalized Sidelobe Canceller

- Problemas:
  - Requiere de tiempo para encontrar los filtros óptimos durante el muestreo.
  - Requiere de calibración de la constante de adaptación para cada diferente SOI.
  - Requiere de una cantidad moderada de micrófonos.

GSC con μ Dinámico

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

mu0 : μ inicial

mu_max : μ máximo

p_x_n : energía de x_n

p_y : energía de y

```
if (mu0*p_x_n/p_y < mu_max)
    mu = mu0/p_y;
else
    mu = mu0/p_x_n;
end
```

Un ejemplo completo se puede observar en gsc_dyn

GSC con μ Dinámico

- Beneficios:
  - Mayor flexibilidad a señales de interés complejas (como voz).
- Problemas:
  - Dos variables a calibrar (mu0 y mu_max).

Siguiente Clase:

Separación de Fuentes por Análisis Estadístico