Estimación de *Múltiples* Direcciones de Arribo

(Multi-DOA)

Preámbulo

- Este tema involucra conceptos profundos, que normalmente se ven en materias como:
  - Algebra lineal
  - Estadística
- Vamos a darle una revisada justamente suficiente para que puedan desarrollar estas técnicas de Multi-DOA en JACK, no más.

Preámbulo

- Por lo tanto, no voy a profundizar en las bases que involucran estas técnicas.
  - Esto no es un estudio extensivo de las bases.
- Les recomiendo darle una revisada a estos conceptos ya sea:
  - Por asesoría
  - Por lectura independiente

Técnicas de Multi-DOA

- “Parches” con Correlación Cruzada.
- MUltiple SIgnal Classification (MUSIC).
- Basado en Beamforming.

4to tipo de Técnicas de Multi-DOA

- Aprendizaje Automático/Profundo***
  - No están diseñados para trabajar en línea.
  - Se “aprenden” la geometría del arreglo.
    - Si hay cambio en la geometría, se requiere re-entrenar.
  - Son “caja negra”, poco explicables.

Nos quedamos con estos 3 tipos de Técnicas de Multi-DOA

- “Parches” con Correlación Cruzada.
- MUltiple SIgnal Classification (MUSIC).
- Basado en Beamforming.

Técnica:

“Parches” con Correlación Cruzada

Correlación Cruzada

- ¿Que no dijimos que era para una sóla fuente?
  - Sí, pero podemos hacer un poco de trampa.

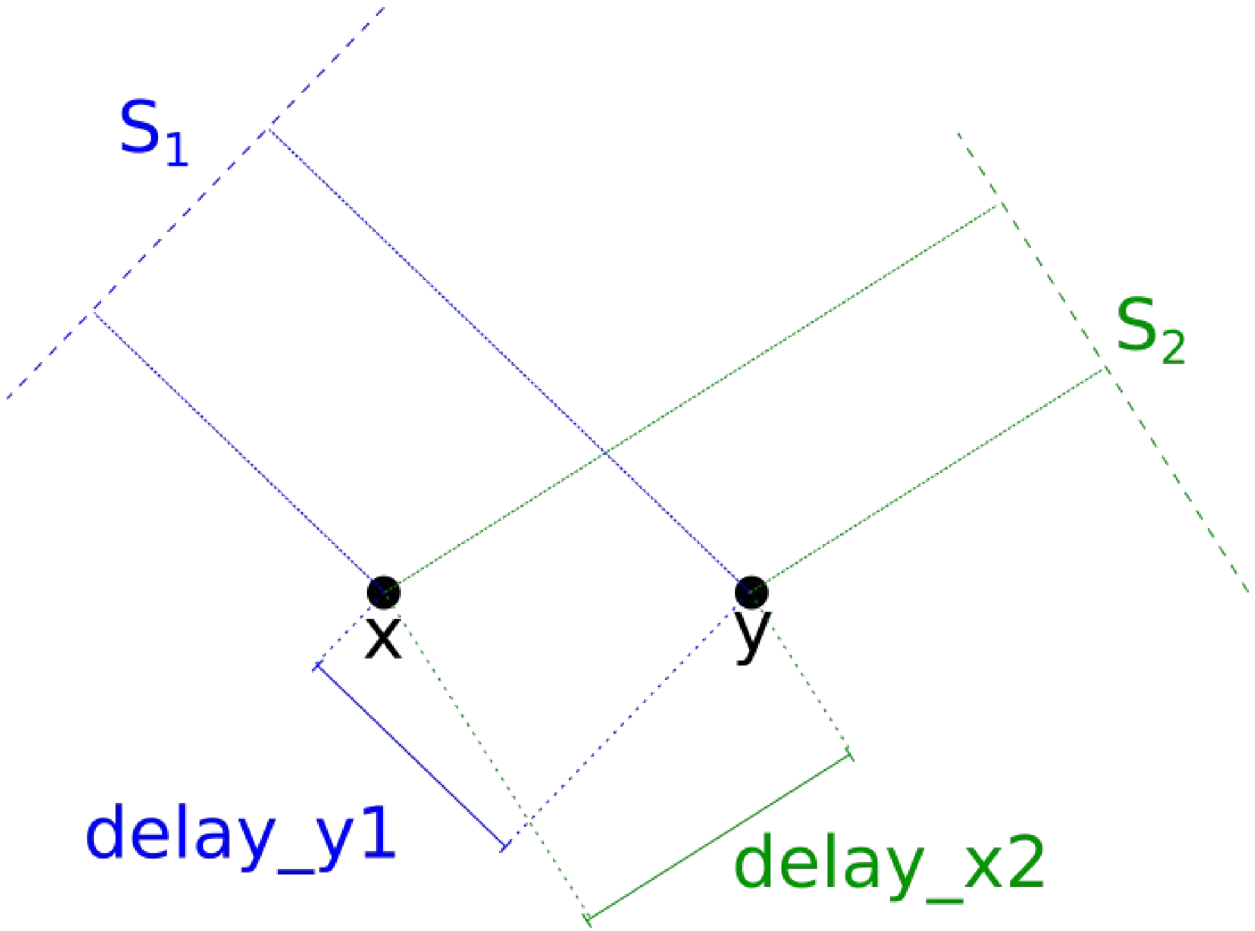
Correlación Cruzada

- Para visualizar esta “trampa”, descarguen al mismo directorio:
  - multigcc.m
  - add_reverb.m
- Antes de correr octave, cambien de directorio a donde los descargaron.
  - Un archivo con terminación “.m” se convierten en un comando/función disponible para octave.
  - El script multigcc manda a llamar la función add_reverb.
- Corran octave, y luego corran multigcc dentro de octave.

multigcc

- Este script crea dos señales de origen.
  - Simulando las señales que emitieron las fuentes.
  - Se llaman “s1” y “s2”.
- Después, desfaza cada señal de origen, les mete ruido y reverberación, y las suma.
  - Así simula las señales capturadas por los micrófonos.
  - Se llaman “x” y “y”.
- Al final, calcula el CCV con Pearson y con GCC-PHAT, de forma de comparar ambas técnicas.

multigcc



multigcc

- También, crea tres figuras:
  - Figura 1: muestra las señales “x” y “y”.
  - Figura 2: el CCV basado en Pearson.
  - Figura 3: el CCV basado en GCC-PHAT.
- Si quieren ver otras señales, pueden crear otra figura con:
  - figure()
  - Y luego mandar a llamar la función plot.

multigcc

- Es posible modificar a multigcc por medio de abrir el archivo multigcc.m (es un archivo de texto).
- Así pueden cambiar el valor de:

  noise_w: qué tanto ruido (0: nada, 1: mucho)

  reverb_w: qué tanta reverberación (0: nada, 1: mucho)

- Recuerden volver a correr multigcc dentro de octave al hacer cualquier cambio.

multigcc

- Prueben con:

  noise_w = 0

  reverb_w = 0

- ¿Que ven en las Figuras 2 y 3?

multigcc

- Prueben con:

  noise_w = 0

  reverb_w = 0

- ¿Que ven en las Figuras 2 y 3?

- Con Pearson: aparecen picos adicionales y son más anchos.
- Con GCC-PHAT: los picos son más angostos, pero no están tan presentes.

multigcc

- Prueben con:

  noise_w = 0

  reverb_w = 0.35

- ¿Que ven en las Figuras 1, 2 y 3?

multigcc

- Prueben con:

  noise_w = 0

  reverb_w = 0.35
- ¿Que ven en las Figuras 1, 2 y 3?

- Las señales “capturadas” están mas gordas.
- Con Pearson: un pico casi desaparece.
- Con GCC-PHAT: los picos siguen presentes, aunque aparecen otros, pero ninguno más grande que los dos picos que andamos buscando.

multigcc

- Prueben con:

  noise_w = 0.35

  reverb_w = 0

- ¿Que ven en las Figuras 1, 2 y 3?

multigcc

- Prueben con:

  noise_w = 0.35

  reverb_w = 0

- ¿Que ven en las Figuras 1, 2 y 3?

- Las señales “capturadas” están algo ruidosas.
- Con Pearson: casi no hay diferencia como cuando lo corrimos sin ruido y sin reverberación.
- Con GCC-PHAT: el pico más pronunciado ya está mas pequeño, el otro pico de interés se ha desparacido y una GRAN cantidad de más picos salieron.

Conclusiones de multigcc

- Cada técnica tiene un pico “preferido” diferente al otro.
  - Observen las señales de origen “s1” y “s2”.
    - Una es más ancha que la otra.

Conclusiones de multigcc

- Pearson calcula la correlación basada en el producto punto, que está relacionada con la cantidad de la área bajo la curva que ambas señales comparten.
  - Por lo tanto, prefiere picos coincidentes anchos.
- GCC-PHAT, por quitar la magnitud en el dominio de la frecuencia, le pone más atención a los cambios fuertes (positivos y negativos) que ocurren a la vez en ambas señales.
  - Por lo tanto, prefiere picos coincidentes angostos.

Conclusiones de multigcc

- Pearson es más robusto ante el ruido que GCC-PHAT.
  - De nuevo, GCC-PHAT es más sensible a cambios en unísono, aun cuando son pequeños.
  - Estas correlaciones “pequeñas” son los picos que el ruido produce.
  - Corranlo de nuevo y verán que los picos de GCC-PHAT cambia cada vez.
    - El ruido es creado con números al azar.
    - Esto no ocurre con Pearson.

Conclusiones de multigcc

- GCC-PHAT es más robusto ante la reverberación que Pearson.
  - Resultado de haber quitado la magnitud en el dominio de la frecuencia.
    - Mientras que ambas señales hayan sido afectadas por la misma reverberación, los cambios fuertes serán iguales en ambas señales.
  - El área conjunta bajo la señal cambia, y si una señal es “tapada”, Pearson no la va a “ver”.

¿Entonces, cual es la trampa?

- Suponiendo que:
  - No hay mucho ruido.
  - No hay mucha reverberación.
- Podemos escoger los picos más altos y presentarlos como los desfases de las señales.

Trampa

- Pero, ¿cómo sabemos cuantos picos escoger?
  - Podemos suponer saberlo.
- Si no, otra forma de escogerlos es presentar todos los picos que estén arriba de un cierto umbral de correlación.

Trampa

- Aún así, habrán veces que ciertos picos no tengan un valor mayor al umbral, y se perderían.
- Podemos ir “acumulando” los picos, y sólo presentar los resultados tras una cierta cantidad de ventanas de tiempo.
- Los picos con alta correlación que hayan aparecido más veces son los que presentamos.

Trampa

- Pero, ¿tras cuántas ventanas presentamos los resultados?
- Y, ¿cuantas veces que hayan aparecido un pico lo consideramos para que lo presentemos como resultado?

- Como dije: es trampa, y no es una solución limpia.
  - Pero tiene sus usos.

Técnica:

MUltiple SIgnal Classification (MUSIC)

MUSIC

- Desarrollado por Ralph Schmidt en 1986:
  - Schmidt, R.O, "Multiple Emitter Location and Signal Parameter Estimation," IEEE Trans. Antennas Propagation, Vol. AP-34 (March 1986), pp.276-280.
  - Lo pueden descargar de la página del curso.
- Y la explicación de este método requiere de varios conceptos siendo explicados primero.

Conceptos Necesarios para MUSIC

- Vectores de Direction (*direction vectors*).
  - Y modelo de señales capturadas
- Eigenvectores y eigenvalores.
- Matriz de Covariancia.

¿Listos?

Vectores de Dirección
(*Direction Vectors*)

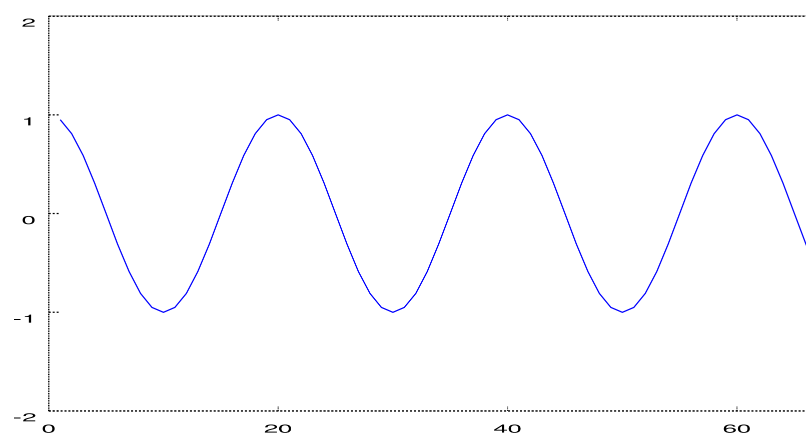
Y

Modelo de Señales Capturadas

Señales Utilizadas para Ejemplos

- Para efectos de este tema, vamos a asumir que las señales que estamos manejando son **señales de frecuencia única**.
- Por lo que en algún momento tendremos que **generalizar esto a todas las frecuencias**.
  - Pero, comencemos sencillo.

$$g(t) = \sin(2\pi f t)$$



Modelo de las Señales Capturadas

- Por la forma en que funcionan el producto entre dos matrices, se puede hacer lo siguiente:

$$\boldsymbol{A}_f = \begin{bmatrix} e^{-i2\pi f T_{1:1}} & e^{-i2\pi f T_{1:2}} & \cdots & e^{-i2\pi f T_{1:D}} \\ e^{-i2\pi f T_{2:1}} & e^{-i2\pi f T_{2:2}} & \cdots & e^{-i2\pi f T_{2:D}} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-i2\pi f T_{M:1}} & e^{-i2\pi f T_{M:2}} & \cdots & e^{-i2\pi f T_{M:D}} \end{bmatrix}$$

$$\boldsymbol{S}_f = \begin{bmatrix} s_1(f) \\ s_2(f) \\ \vdots \\ s_D(f) \end{bmatrix}$$

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

$$\boldsymbol{S} = \begin{bmatrix} s_1(1) & s_1(2) & \cdots & s_1(N) \\ s_2(1) & s_2(2) & \cdots & s_2(N) \\ \vdots & \vdots & \ddots & \vdots \\ s_D(1) & s_D(2) & \cdots & s_D(N) \end{bmatrix}$$

Donde:

$s_d(f)$: es una señal de origen en frecuencia f

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

A_f: matriz que contiene los vectores de dirección *para la frecuencia f*

X_f: señales capturadas *en la frecuencia f*; cada renglón representa un micrófono

S_f: señales de origen *en la frecuencia f*; cada renglón representa una señal de origen

Modelo de las Señales Capturadas

- Por la forma en que funcionan el producto entre dos matrices, se puede hacer lo siguiente:

$$\boldsymbol{A_f}=\begin{bmatrix} e^{-i2\pi f T_{1:1}} & e^{-i2\pi f T_{1:2}} & \cdots & e^{-i2\pi f T_{1:D}} \\ e^{-i2\pi f T_{2:1}} & e^{-i2\pi f T_{2:2}} & \cdots & e^{-i2\pi f T_{2:D}} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-i2\pi f T_{M:1}} & e^{-i2\pi f T_{M:2}} & \cdots & e^{-i2\pi f T_{M:D}} \end{bmatrix} \qquad \boldsymbol{S_f}=\begin{bmatrix} s_1(f) \\ s_2(f) \\ \vdots \\ s_D(f) \end{bmatrix}$$

$$\boldsymbol{X_f}=\boldsymbol{A_f}\boldsymbol{S_f}$$

Lo que nos interesa estimar.

Donde:

$s_d(f)$: es una señal de origen en frecuencia f

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

A_f: matriz que contiene los vectores de dirección *para la frecuencia f*

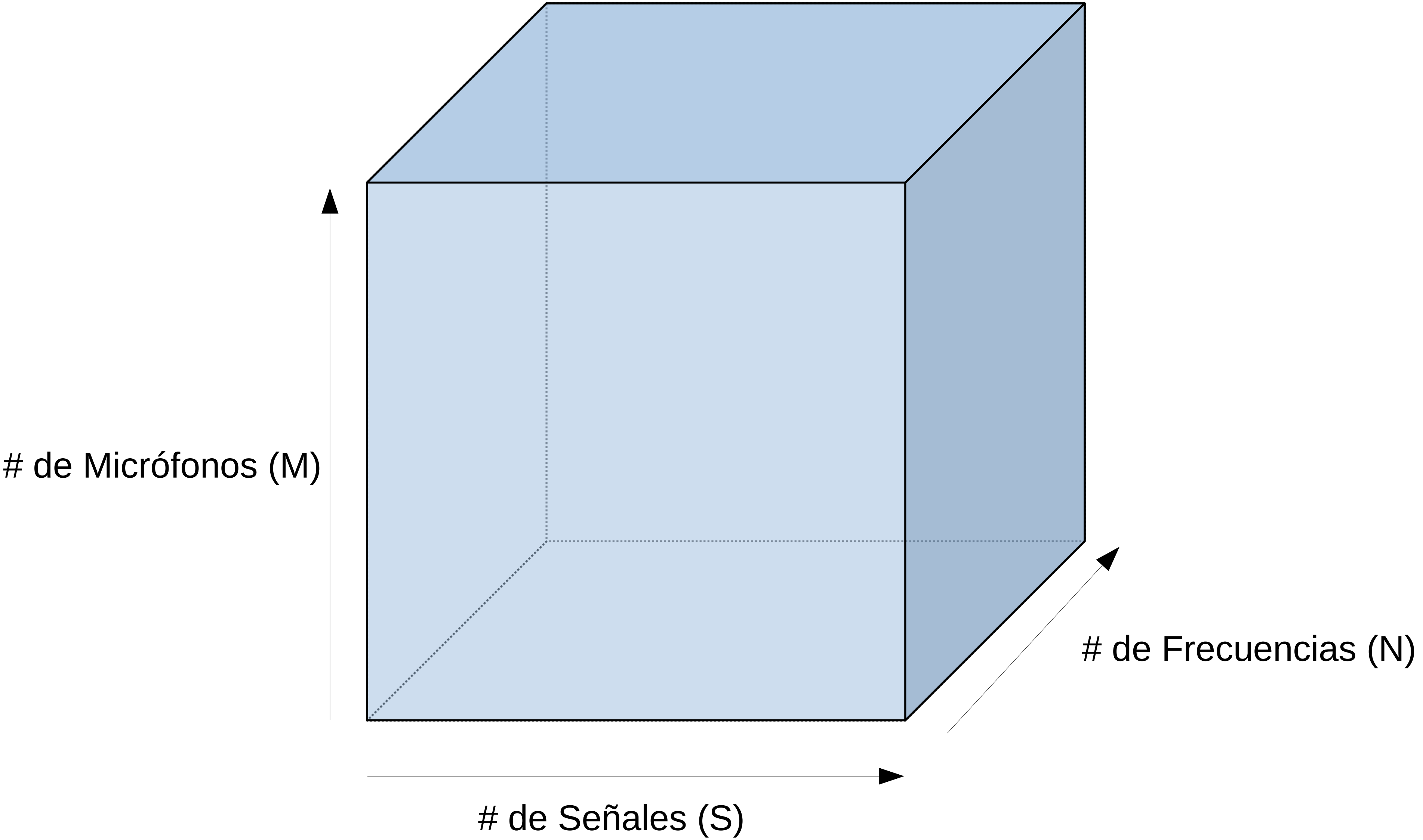
X_f: señales capturadas *en la frecuencia f*; cada renglón representa un micrófono

S_f: señales de origen *en la frecuencia f*; cada renglón representa una señal de origen

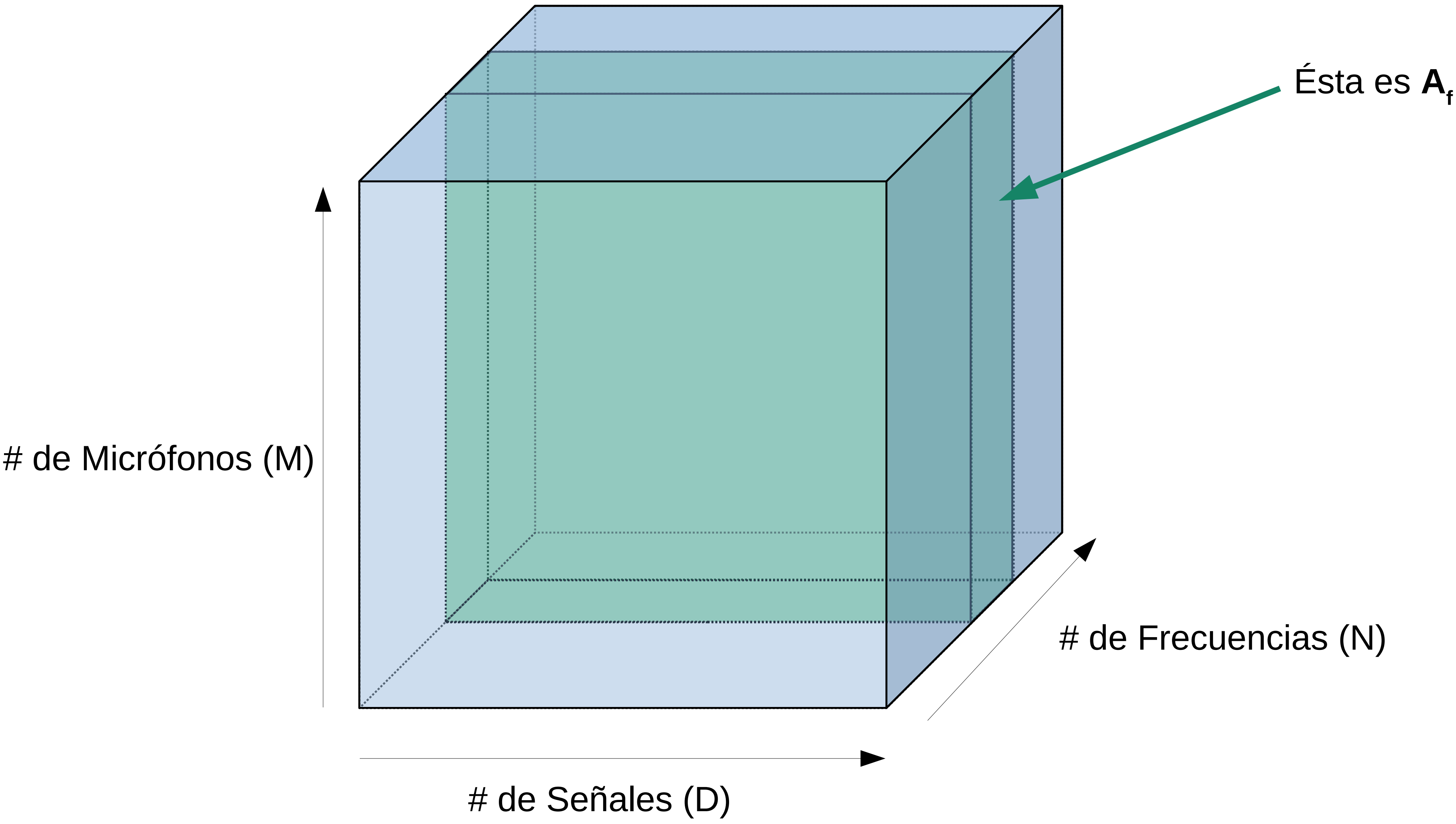
¡OJO!

- La matriz A_f es para una sola frecuencia.
- De hecho, A_f es parte de un tensor de 3 dimensiones:

Tensor de Direction Vectors



Tensor de Direction Vectors



Modelo de las Señales Capturadas

- Tatúenselo en la mente:

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

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

Donde:

$s_d(f)$: es una señal de origen en frecuencia f

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

A_f: matriz que contiene los vectores de dirección *para la frecuencia f*

X_f: señales capturadas *en la frecuencia f*; cada renglón representa un micrófono

S_f: señales de origen *en la frecuencia f*; cada renglón representa una señal de origen

Direction Vector

- Enlista los desfases que sufre la señal de origen en cada micrófono.
- Se establece algún micrófono como “referencia” del cual se calculan los desfases relativos.
  - Normalmente es el primer micrófono.
- El direction vector para una señal “d”, para la frecuencia “f_1”, con M micrófonos:

$$\boldsymbol{A}_{\boldsymbol{d}:\boldsymbol{f}_1}=\begin{bmatrix} 1 \\ e^{-i2\pi f_1 T_{2:d}} \\ e^{-i2\pi f_1 T_{3:d}} \\ \vdots \\ e^{-i2\pi f_1 T_{M:d}} \end{bmatrix}$$

Cálculo de Desfases para un Direction Vector

- Se asume que se conoce su dirección de arribo.
- Para encontrar el desfase que sufre del micrófono de diferencia, se puede utilizar el inverso del modelo del campo lejano:

$$t = d \sin(\theta) / V_{sound}$$

t: desfase entre micrófonos de la señal de interés

d: distancia entre micrófonos

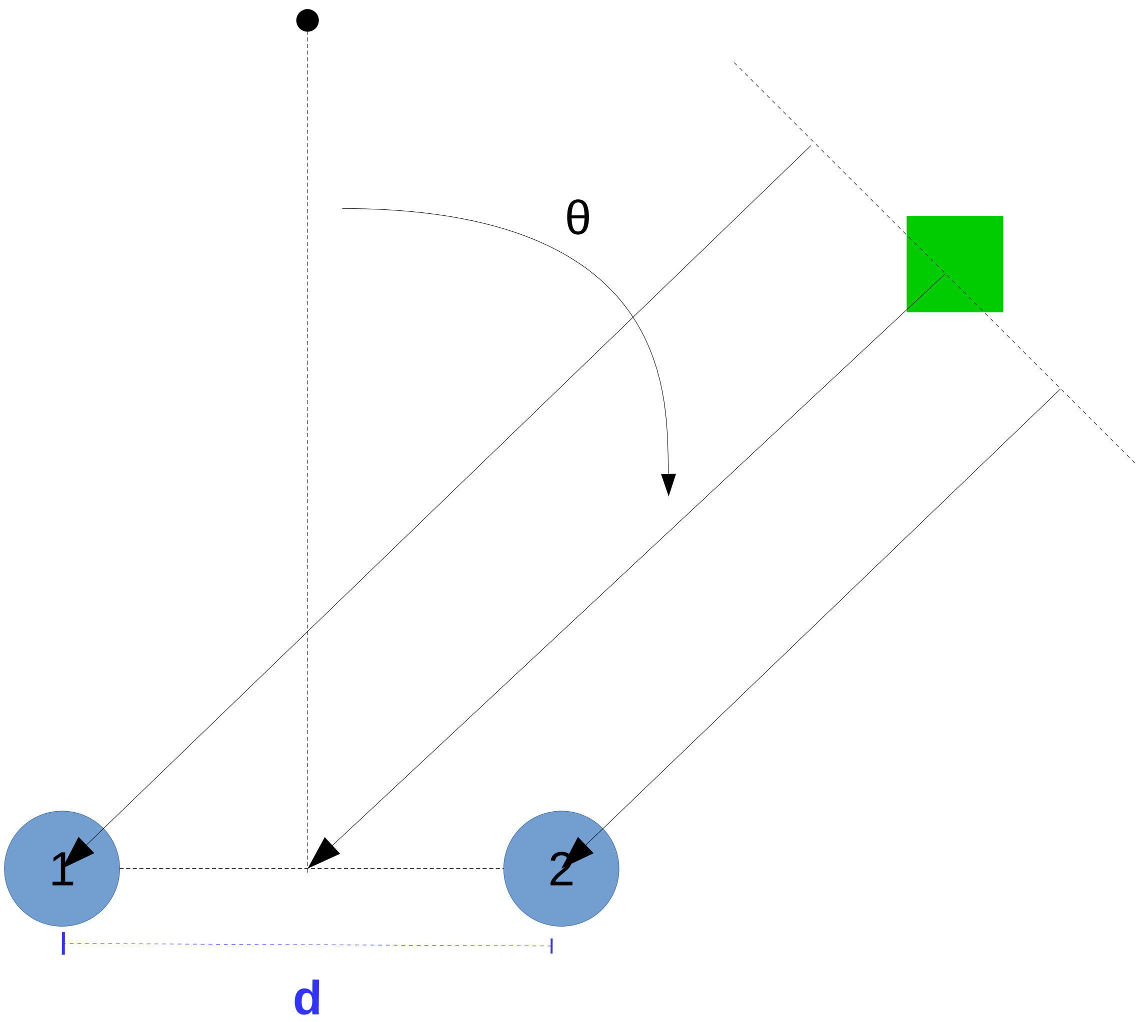
V_{sound}: velocidad del sonido

θ: dirección de arribo de la señal

- ¿Cómo se mide el ángulo?

Para dos micrófonos

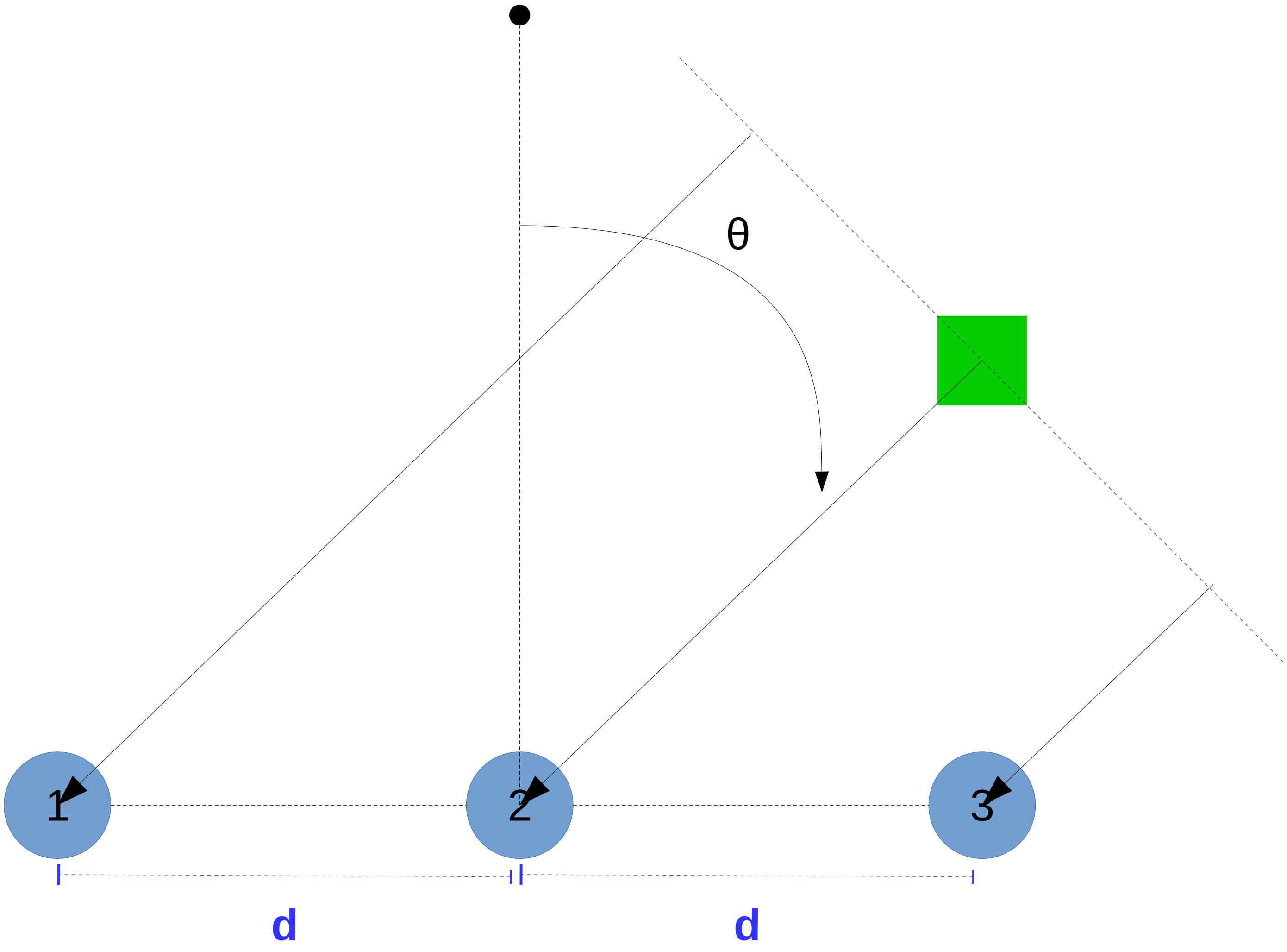
$$t_{2-1} = \frac{\sin(\theta) d}{V_{sound}}$$



Para tres micrófonos

$$t_{2-1}=\frac{\sin(\theta)d}{V_{sound}}$$

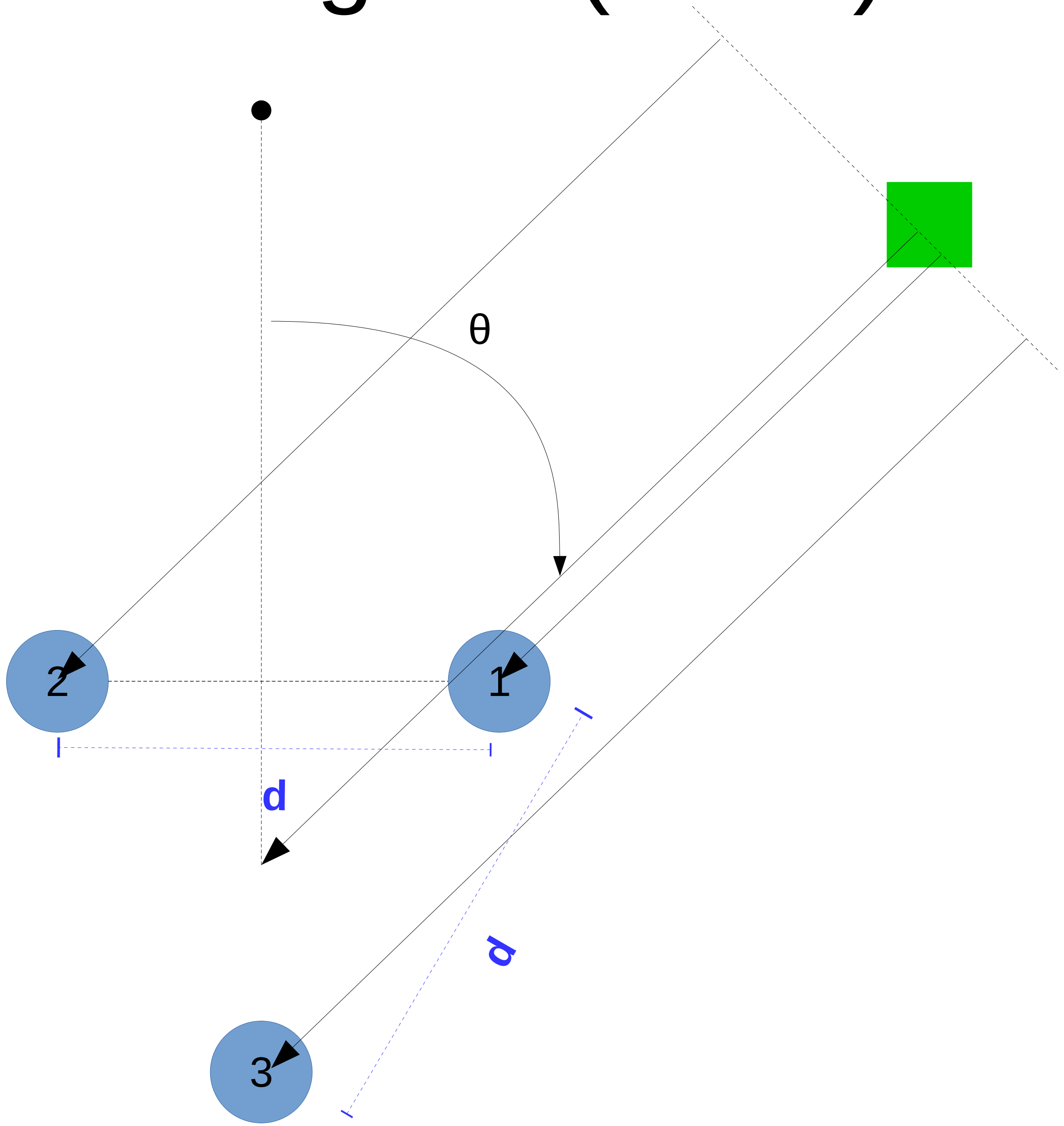
$$t_{3-1}=\frac{\sin(\theta)d*2}{V_{sound}}$$



Para tres micrófonos en un arreglo triangular (AIRA)

$$t_{2-1} = -\frac{\sin(\theta)d}{V_{sound}}$$

$$t_{3-1} = ???$$

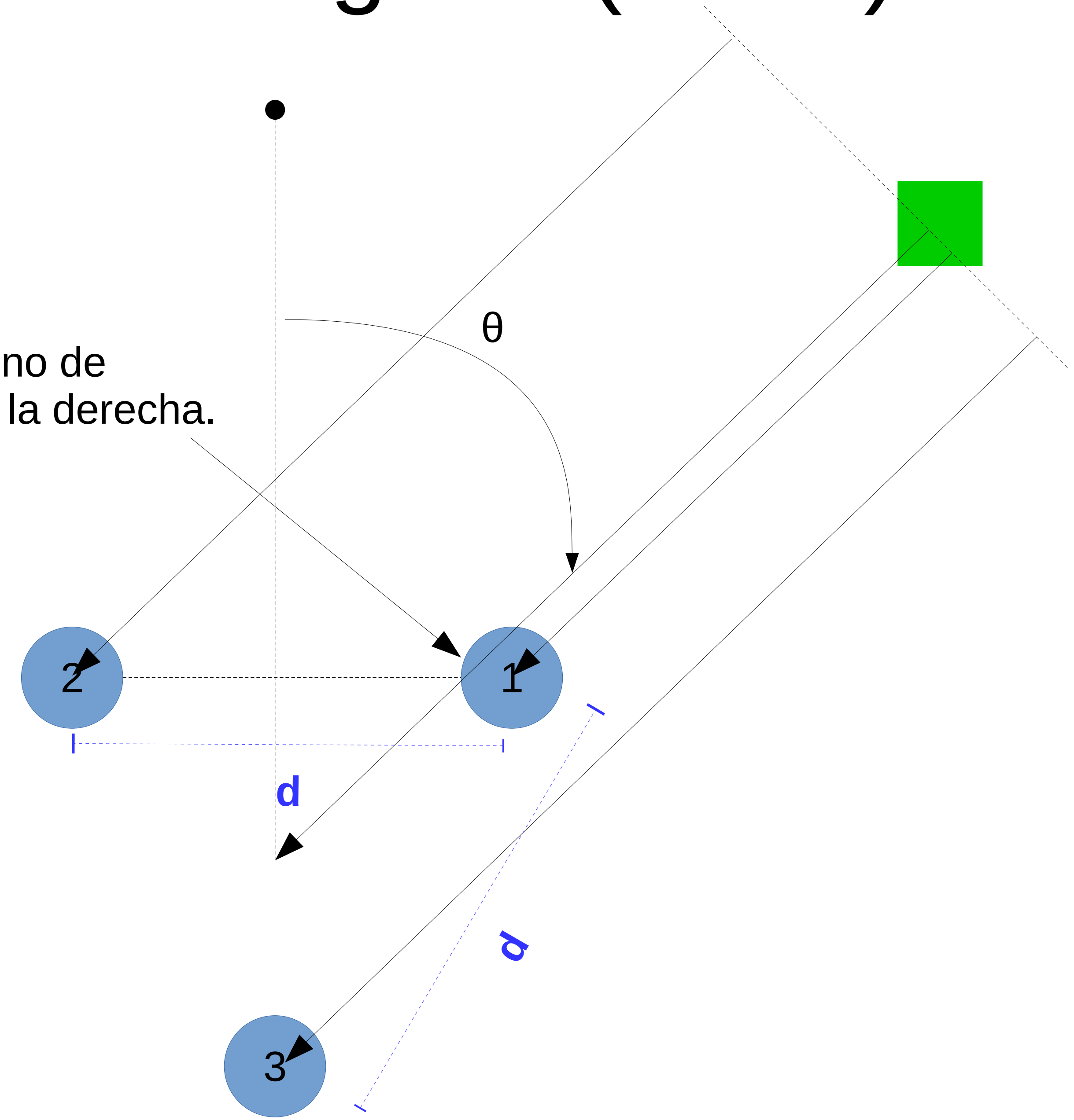


Para tres micrófonos en un arreglo triangular (AIRA)

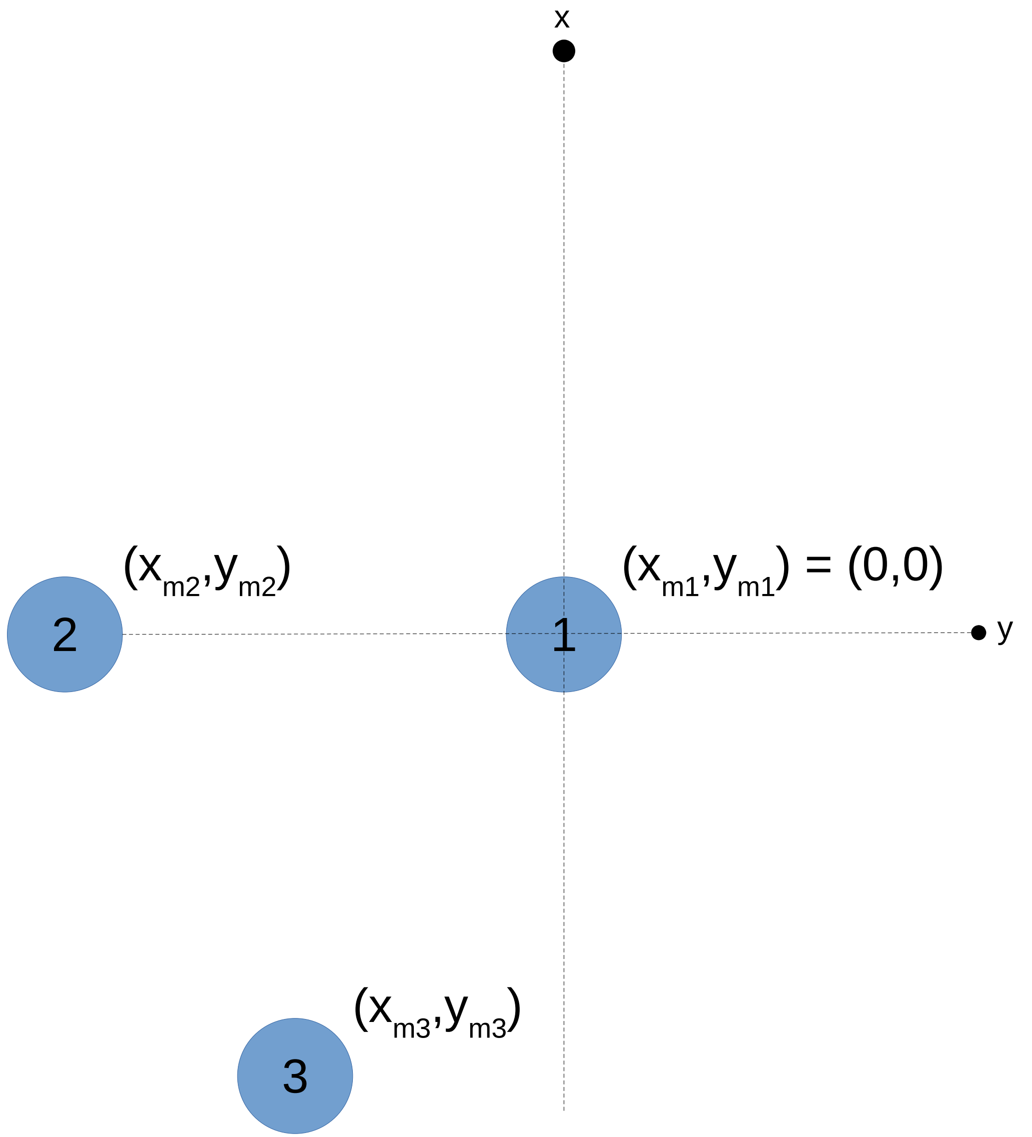
$$t_{2-1} = -\frac{\sin(\theta)\, d}{V_{sound}}$$

$$t_{3-1} = ???$$

OJO:
En AIRA el micrófono de referencia es el de la derecha.



Para tres micrófonos en un arreglo triangular (AIRA)

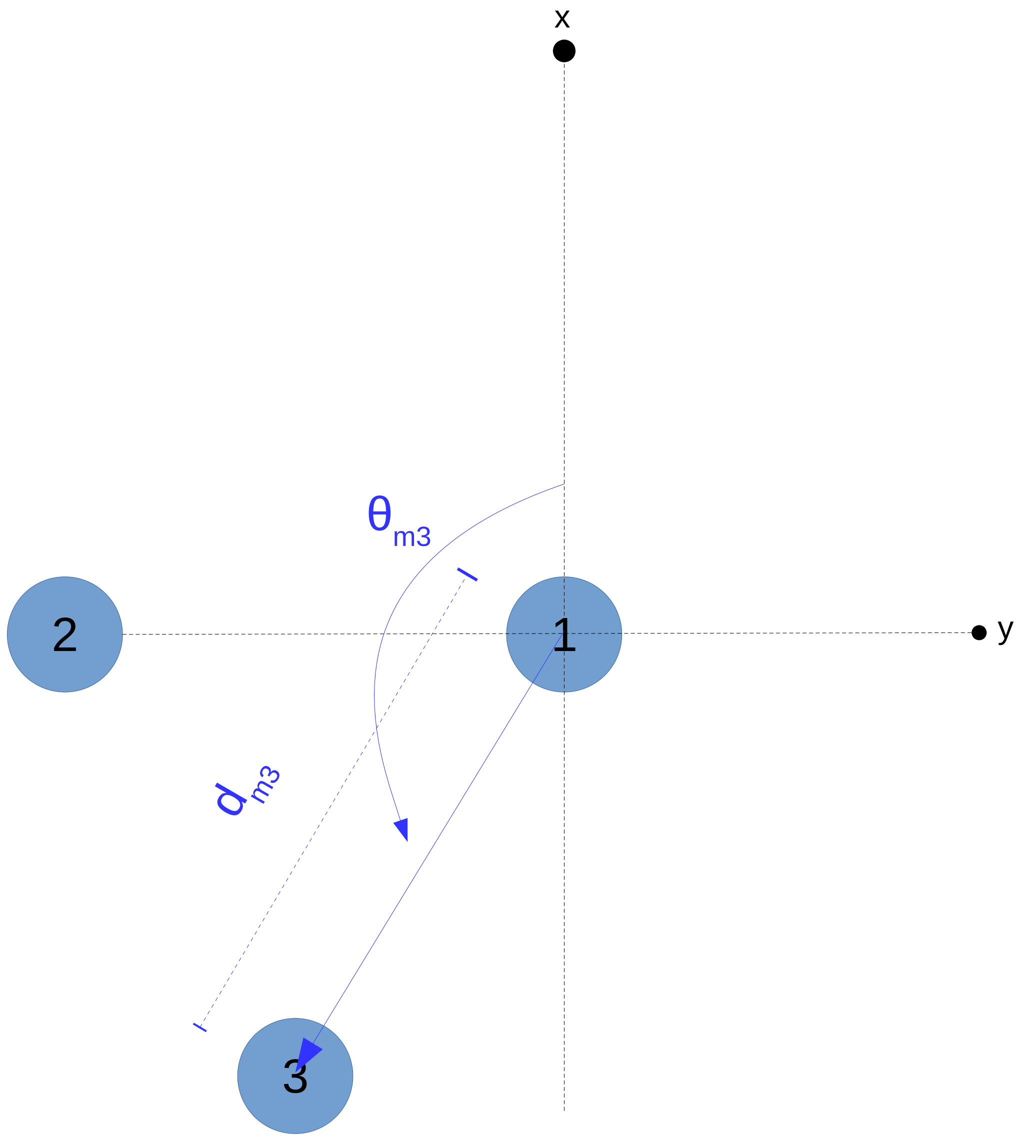


Para tres micrófonos en un arreglo triangular (AIRA)

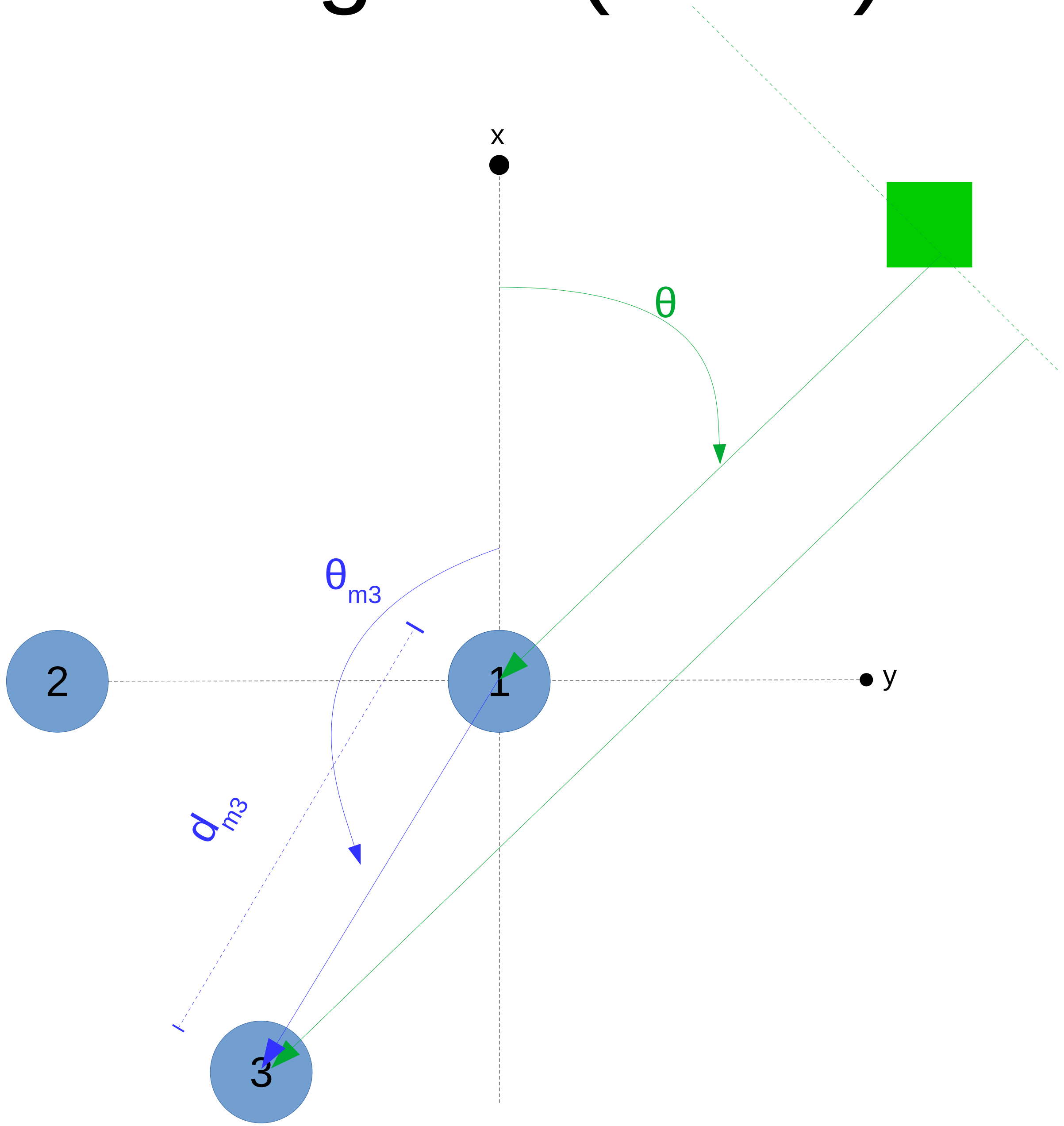
$$x_{m1}=y_{m1}=0$$

$$\theta_{m3}=atan\left(\frac{y_{m3}}{x_{m3}}\right)$$

$$d_{m3}=\sqrt{y_{m3}^2+x_{m3}^2}$$



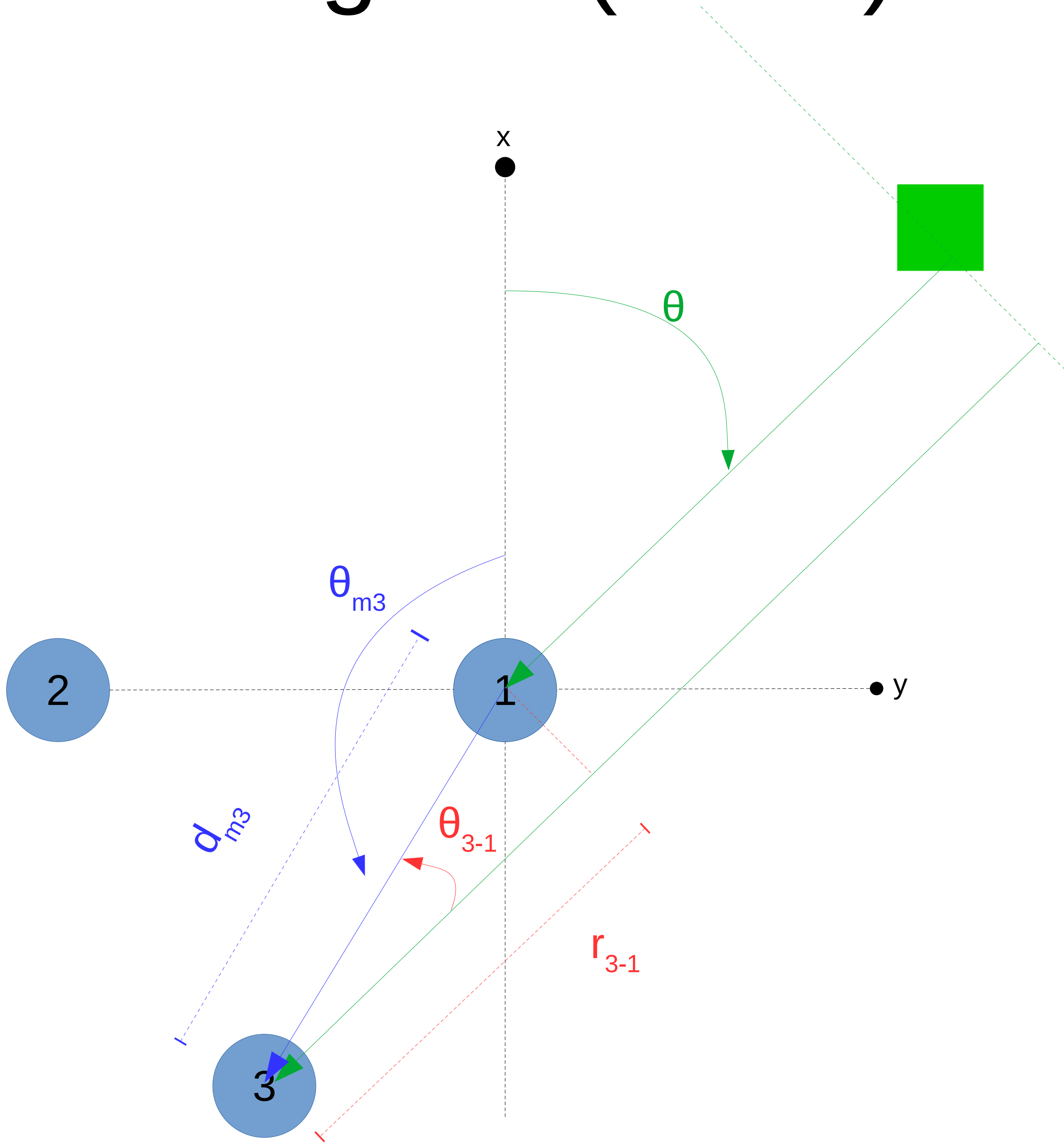
Para tres micrófonos en un arreglo triangular (AIRA)



Para tres micrófonos en un arreglo triangular (AIRA)

$$r_{3-1}=d_{m3}*\cos(\theta_{3-1})$$

$$\theta_{3-1}=?$$

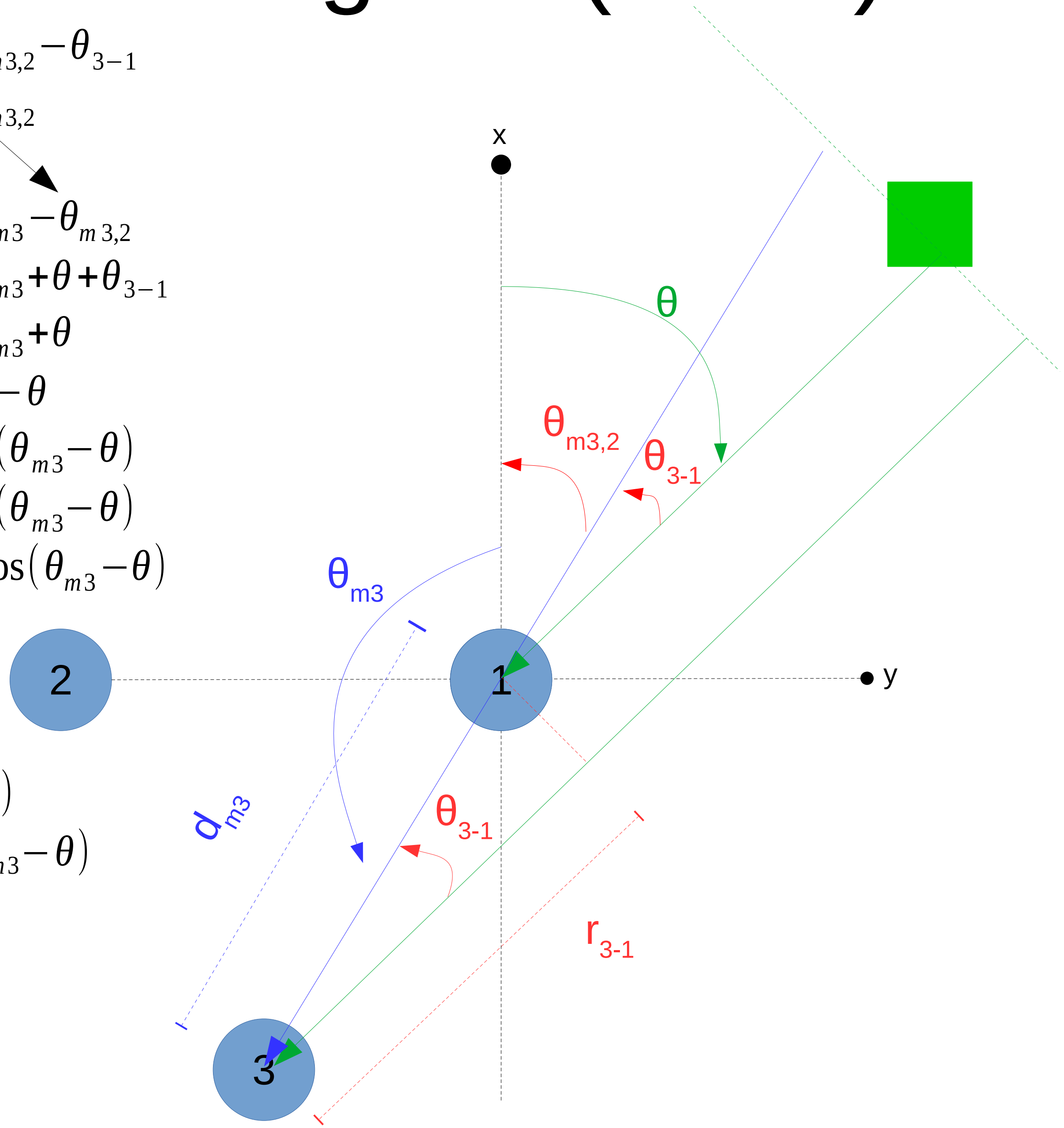


Para tres micrófonos en un arreglo triangular (AIRA)

$$
\begin{aligned}
\theta &= -\theta_{m3,2} - \theta_{3-1}\\
\theta + \theta_{3-1} &= -\theta_{m3,2}
\end{aligned}
$$

$$
\begin{aligned}
-180 &= -\theta_{m3} - \theta_{m3,2}\\
-180 &= -\theta_{m3} + \theta + \theta_{3-1}\\
-180 - \theta_{3-1} &= -\theta_{m3} + \theta\\
180 + \theta_{3-1} &= \theta_{m3} - \theta\\
\cos(180 + \theta_{3-1}) &= \cos(\theta_{m3} - \theta)\\
-\cos(\theta_{3-1}) &= \cos(\theta_{m3} - \theta)\\
\cos(\theta_{3-1}) &= -\cos(\theta_{m3} - \theta)
\end{aligned}
$$

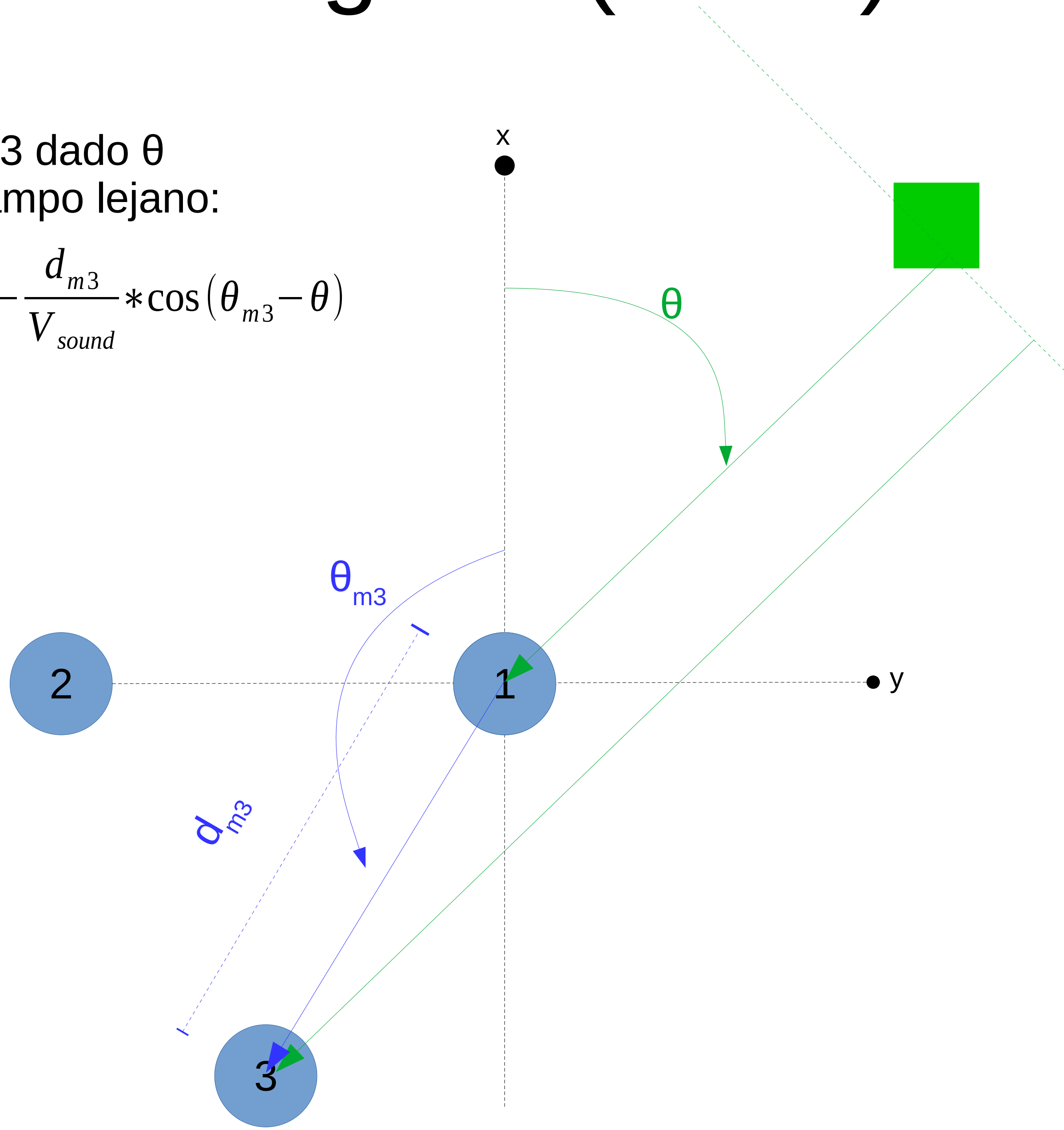
$$
\begin{aligned}
r_{3-1} &= d_{m3} * \cos(\theta_{3-1})\\
r_{3-1} &= -d_{m3} * \cos(\theta_{m3} - \theta)
\end{aligned}
$$



Para tres micrófonos en un arreglo triangular (AIRA)

Desfase en m3 dado θ asumiendo campo lejano:

$$t_{3-1}(\theta)=-\frac{d_{m3}}{V_{sound}}*\cos(\theta_{m3}-\theta)$$



Marcos de Referencia y Modelo de Campo Lejano

- Esta forma de calcular los desfases asume un modelo de campo lejano.
  - Así, el marco de referencia del primero micrófono y el marco de referencia del arreglo de micrófonos comparten los mismos ángulos.
- Si no se cumple esta suposición, ésta forma de calcular desfases no es consistente con la realidad.
- Si sí se cumple, esta forma de cálculo es **generalizable** a cualquier posición de micrófono.

Eigenvectores y Eigenvalores

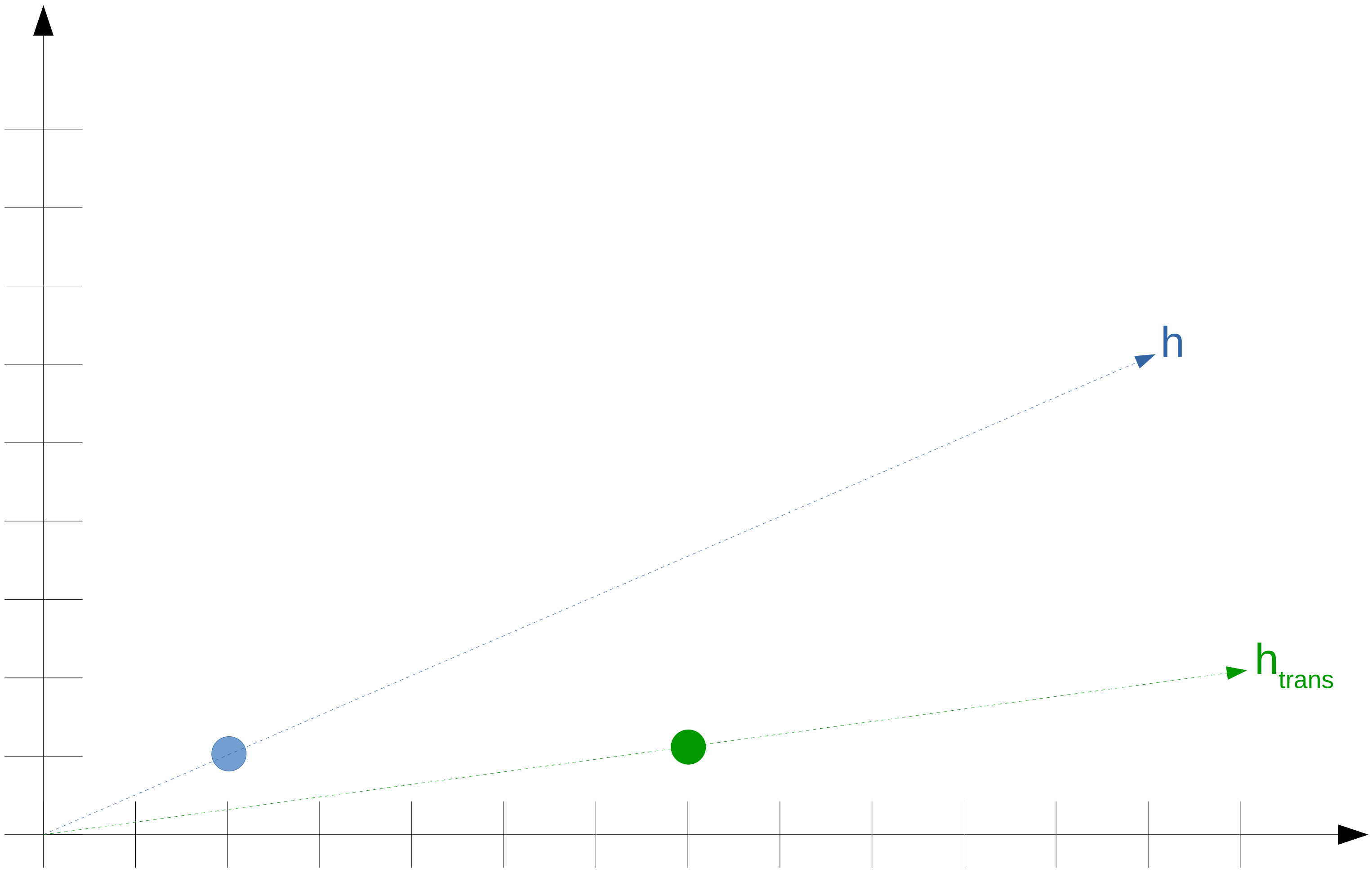
La idea es...

- Imaginen una matriz que normalmente es utilizada para transformar vectores.
- Por ejemplo, digamos que tenemos una matriz de transformación B, y un vector al que queremos transformar h a h_{trans}:

$$B=\begin{bmatrix}1 & 0\\ 1 & 3\end{bmatrix} \qquad h=\begin{bmatrix}1\\ 2\end{bmatrix}$$

$$h_{trans}=Bh=\begin{bmatrix}1 & 0\\ 1 & 3\end{bmatrix}\begin{bmatrix}1\\ 2\end{bmatrix}=\begin{bmatrix}1\\ 7\end{bmatrix}$$

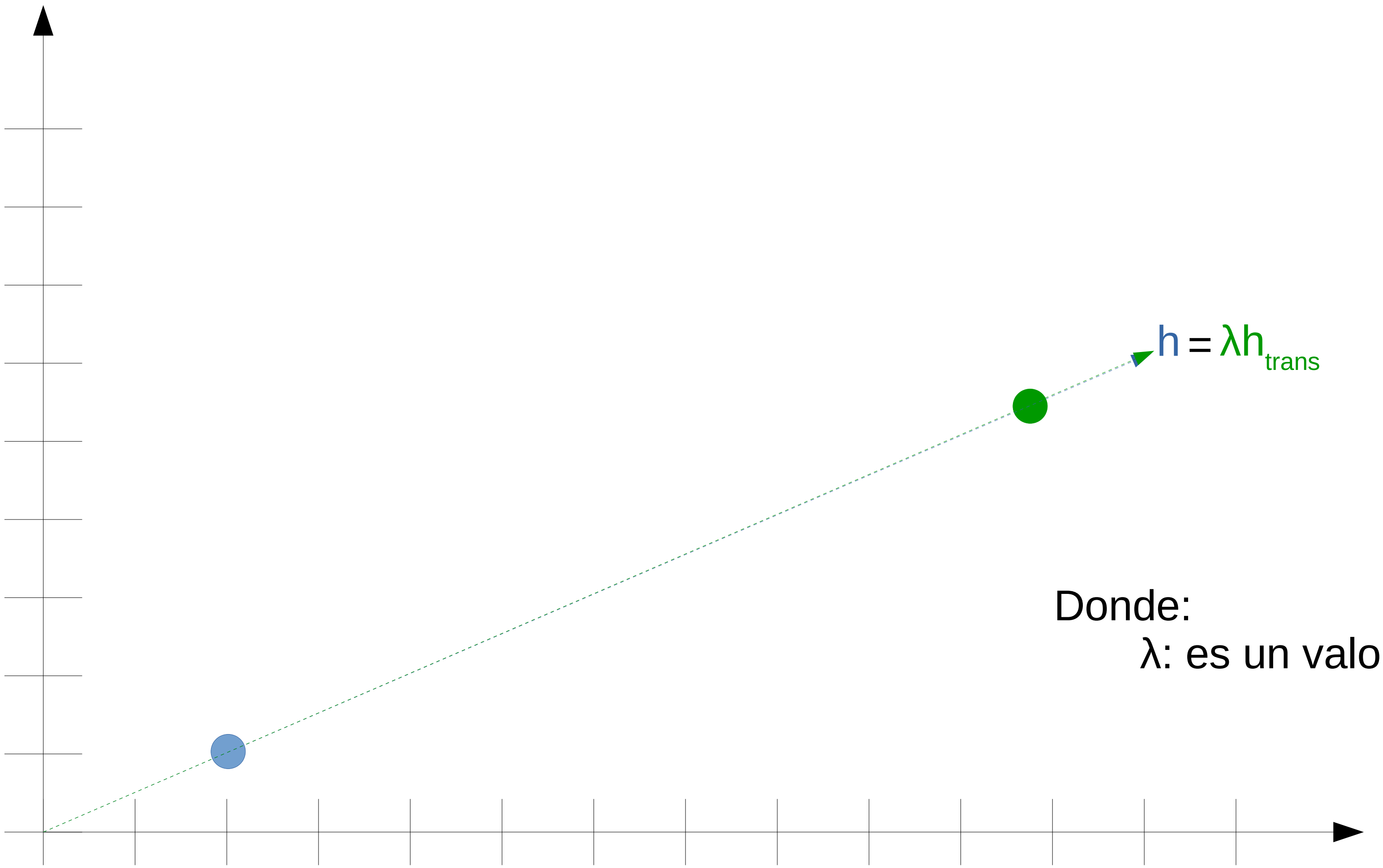
Este es el efecto de B en h



Pero...

- Hay vectores excepcionales que no les ocurre esto.
- Que al multiplicarlos por alguna B, su dirección no cambia.

Y sucede algo así



Donde:

λ: es un valor escalar

Eigenvector y Eigenvalores

- Estos vectores son los Eigenvectores de esa matriz.
  - También conocidos como “Vectores Característicos”.
- Y los valores λ que les corresponde, son sus Eigenvalores.
  - También conocidos como “Valores Característicos”.
- Dícese, v es un eigenvector de B, y λ su eigenvalor correspondiente si se cumple:

$$Bv = \lambda v$$

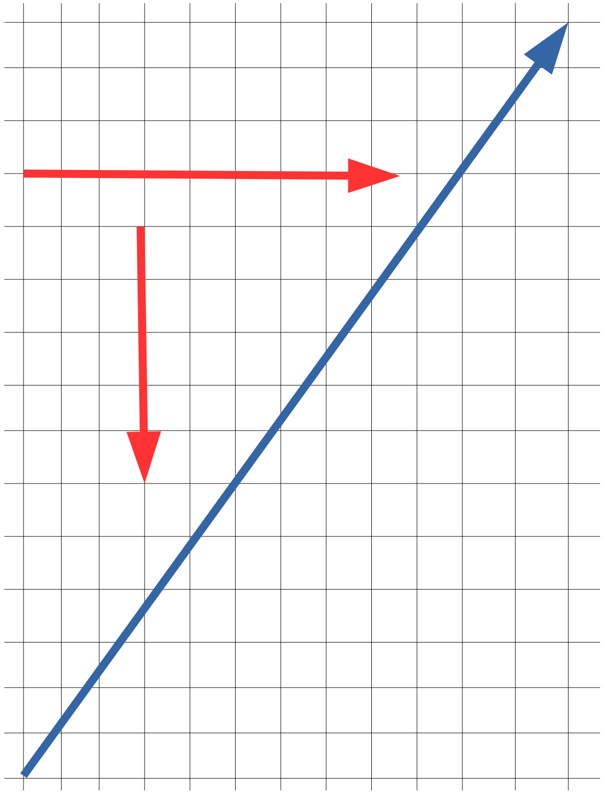
Y, ¿luego?

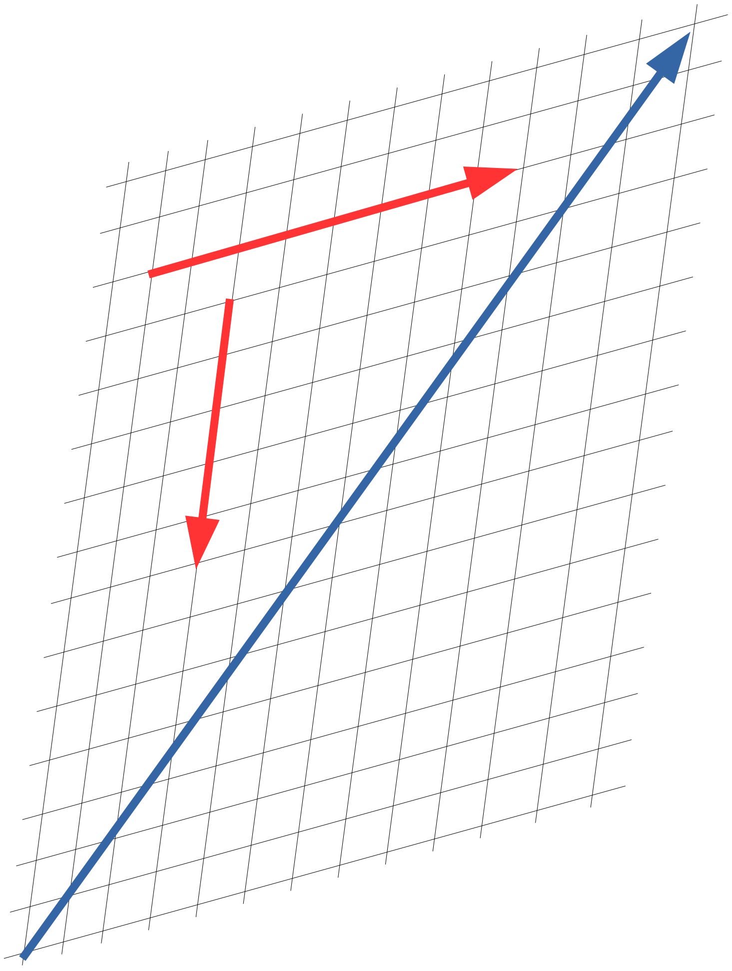
- Imaginen un pedazo de tela.
  - Cada nudo de la tela actúa como un posible vector.
- Ahora imaginen que estiran esa tela.
  - Esta acción es como aplicar la matriz B a ese conjunto de vectores.

Algo así...

Los vectores *azules*, aun cuando inician y terminan en el mismo punto de ambos planos cartesianos, tienen el mismo ángulo. Su única diferencia es en la magnitud.

Es un **eigenvector**, y la escala de la diferencia de magnitud es el **eigenvalor**.





Otra manera de verlo...

- Imaginen que ustedes tienen un líquido misterioso del cual desconocen de qué está compuesto.
- Al combinarlo con otras substancias (como agua), observan que hay reacciones químicas (burbujas, cambio de color, etc.).
- Pero también obervan que no hay reacción química con una substancia en específico (una *eigen-substancia*).
- Esto significa que es probable que el líquido misterioso esta compuesto, en parte, por esta eigen-substancia.

Entonces...

- Los eigenvectores describen, de cierta manera, las tendencias numéricas internas de la matriz.
- De hecho, la palabra “eigen” en alemán significa: “inherente, característico, propio”.
  - También significa “peculiar”.

Háganme el bendito favor...

- Pero, esto es importante recordarlo, ya que es un elemento crítico del algoritmo MUSIC.

Propiedades de Eigenvectores/Eigenvalores

- Los vectores y los valores son correspondientes.
  - Un eigenvalor por eigenvector.
- Se pueden calcular los eigenvectores/eigenvalores de una matriz, siempre y cuando ésta sea **cuadrada**.
  - Tantos renglones como columnas.
- La cantidad de eigenvectores/eigenvalores que contiene una matriz es el número de columnas o renglones.
  - B tiene 2 eigenvectores/eigenvalores.

$$B=\begin{bmatrix} 1 & 0 \\ 1 & 3 \end{bmatrix}$$

Propiedades de Eigenvectores/Eigenvalores

- Los eigenvectores calculados, son ortogonales entre sí.
  - Es decir, si B tiene dos eigenvectores v_1 y v_2.
  - Entonces:

$$v_1 \cdot v_2 = 0$$

  - *Esto va a ser **muy** importante más adelante.*

¿Cómo se calculan?

- Este proceso se le conoce como **eigendescomposición**.
- La idea es descomponer a B de tal forma que:

$$B = V \Lambda V^{-1}$$

$$V = \begin{bmatrix} v_1 & v_2 & \cdots & v_K \end{bmatrix}$$

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_K \end{bmatrix}$$

Donde:

V: es una matriz que contiene por cada columna un eigenvector de B

Λ: una matriz, en la que en su diagonal contenga los eigenvalores de B

K: es el número de columnas o renglones de B, también conocido como **su tamaño**

En Octave

- La función “eig” de octave hace la eigendescomposición:

```
B = [1 0; 1 3]

[e_vectors e_values] = eig(B)
```

En C/C++

- Una implementación popular está dentro la biblioteca de algebra lineal llamado **Eigen**.
  - Está en C++
  - Lo pueden instalar con:

    sudo apt-get install libeigen3-dev
  - Documentación: http://eigen.tuxfamily.org/dox/
  - Requiere añadir la ubicación de sus headers al compilar:

    -I/usr/include/eigen3

Código Ejemplo con Eigen

```
#include <iostream>
#include <Eigen/Eigen>

int main() {
  Eigen::MatrixXd A(2,2);
  A(0,0) = 1; A(0,1) = 2; A(1,0) =  2; A(1,1) =  3;
  std::cout << "The matrix A:\n" << A << std::endl;

  Eigen::EigenSolver<Eigen::MatrixXd> es(A);
  std::cout << "The eigenvalues of A are:\n" << es.eigenvalues() << std::endl;
  std::cout << "The eigenvectors of A are:\n" << es.eigenvectors() << std::endl;
}
```

Compilar con: g++ -I/usr/include/eigen3 ARCHIVO.cpp -o PROGRAMA

Eigen

- Hay mucho más que se puede hacer en Eigen:
  - Aritmética de matrices y vectores.
  - Transposición y conjugación.
  - Producto punto, inversa y determinante.
  - Etc.
- Material para clase: *eigen_examples.cpp*
  - Lo que se muestra en ese código es suficiente para llevar a cabo sus proyectos.
- Pero, la documentación de Eigen es muy completa.

  https://eigen.tuxfamily.org/dox/

Eigen: Números Complejos

- Eigen utiliza la implementación estándar de números complejos de C++:

  std::complex

- Por lo tanto, si se va a utilizar FFTw3, se requiere utilizar la función de *reinterpret_cast* para compatibilidad.
  - Ver jack_fft.cpp en la página del curso.

Covariancia

Recordemos

- Al centrar las señales antes de calcular el coeficiente Pearson, éste se convertía en:
  - La covarianciа entre las señales, dividido entre sus desvaciones estándar.
- Entonces, la covariancia es también una medida de correlación entre las señales.
  - Nada más que sin desfasarlas primero.
  - Y no está normalizada.
  - Pero aún así indica una forma de correlación útil.

La Matriz de Covariancia

- Una forma rápida de calcular la covariancia entre varias señales, es calculando su matriz de covariancia, por medio de:

$$R = XX^H$$

Donde:

R: es la matriz de covariancia

X: es la matriz de las señales capturadas

^{H}: es la operación de transposición conjugada de matrices (Hermitiana)

$$\begin{bmatrix} 1+i1 & 2-i2 \\ 3+i3 & 4-i4 \end{bmatrix}^H = \begin{bmatrix} 1-i1 & 3-i3 \\ 2+i2 & 4+i4 \end{bmatrix}$$

Recordatorio

- X es la matriz de las señales **capturadas**.
  - 1 renglón por micrófono.
- La podemos modelar como: $\boldsymbol{X}_f = \boldsymbol{A}_f \boldsymbol{S}_f$

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

Donde:

$s_d(f)$: es una señal de origen en frecuencia f

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

A_f: matriz que contiene los vectores de dirección *para la frecuencia f*

X_f: señales capturadas *en la frecuencia f*; cada renglón representa un micrófono

S_f: señales de origen *en la frecuencia f*; cada renglón representa una señal de origen

Matriz de Covariancia

- Es una matriz cuadrada de tamaño del número de micrófonos.
- En el caso de 2 micrófonos, tiene la forma de:

$$R=\begin{bmatrix} cov(x_1,x_1) & cov(x_1,x_2) \\ cov(x_2,x_1) & cov(x_2,x_2) \end{bmatrix}$$

Matriz de Covariancia en una Frecuencia

- La covariancia normalmente se calcula con ventanas completas de las señales.
- Pero, es útil utilizar la información de covariancia de sólo una frecuencia para saber como las señales co-varían en dicha frecuencia.

$$\boldsymbol{X}=\begin{bmatrix} x_1(1) & x_1(2) & \cdots & x_1(N) \\ x_2(1) & x_2(2) & \cdots & x_2(N) \\ \vdots & \vdots & \ddots & \vdots \\ x_M(1) & x_M(2) & \cdots & x_M(N) \end{bmatrix}$$

Matriz de Covariancia en una Frecuencia

- Si X está en el dominio de la frecuencia, y queremos calcular la covariancia de la segunda frecuencia (R_{f2}):

$$\boldsymbol{X} = \begin{bmatrix} x_1(1) & \boxed{x_1(2)} & \cdots & x_1(N) \\ x_2(1) & \boxed{x_2(2)} & \cdots & x_2(N) \\ \vdots & \vdots & \ddots & \vdots \\ x_M(1) & \boxed{x_M(2)} & \cdots & x_M(N) \end{bmatrix}$$

$$\downarrow$$

$$\boldsymbol{X}_{f_2} = \begin{bmatrix} x_1(2) \\ x_2(2) \\ \vdots \\ x_M(2) \end{bmatrix} \longrightarrow R_{f_2} = X_{f_2} X_{f_2}^H$$

Matriz de Covariancia en una Frecuencia

- Pero esto sólo utiliza la información de la frecuencia en **un momento en el tiempo**
- Esto no entrega una matriz de covariancia “confiable”, ya que no es posible observar una varianza sin más muestras.
- Por lo tanto, al calcular la matriz de covariancia de una frecuencia es importante que se utilicen varias muestras de dicha frecuencia a lo largo del tiempo.
- Es decir...

Matriz de Covariancia en una Frecuencia

$$\boldsymbol{X}_1=\begin{bmatrix} x_{1:1}(1) & x_{1:1}(2) & \cdots & x_{1:1}(N) \\ x_{1:2}(1) & x_{1:2}(2) & \cdots & x_{1:2}(N) \\ \vdots & \vdots & \ddots & \vdots \\ x_{1:M}(1) & x_{1:M}(2) & \cdots & x_{1:M}(N) \end{bmatrix}$$

$$\boldsymbol{X}_2=\begin{bmatrix} x_{2:1}(1) & x_{2:1}(2) & \cdots & x_{2:1}(N) \\ x_{2:2}(1) & x_{2:2}(2) & \cdots & x_{2:2}(N) \\ \vdots & \vdots & \ddots & \vdots \\ x_{2:M}(1) & x_{2:M}(2) & \cdots & x_{2:M}(N) \end{bmatrix}$$

$$\boldsymbol{X}_3=\begin{bmatrix} x_{3:1}(1) & x_{3:1}(2) & \cdots & x_{3:1}(N) \\ x_{3:2}(1) & x_{3:2}(2) & \cdots & x_{3:2}(N) \\ \vdots & \vdots & \ddots & \vdots \\ x_{3:M}(1) & x_{3:M}(2) & \cdots & x_{3:M}(N) \end{bmatrix}$$

$$\boldsymbol{X}_{f_2}=\begin{bmatrix} x_{1:1}(2) & x_{2:1}(2) & x_{3:1}(2) \\ x_{1:2}(2) & x_{2:2}(2) & x_{3:2}(2) \\ \vdots & \vdots & \vdots \\ x_{1:M}(2) & x_{2:M}(2) & x_{3:M}(2) \end{bmatrix}$$

$$R_{f_2}=X_{f_2}X_{f_2}^{H}$$

Matriz de Covariancia Muestrada
(*Sample Covariance Matrix*)

¿Para qué es útil?

- Ya estamos listos...

MUSIC

(ahora sí)

Premisa de MUSIC

- ¿Que sucedería si le sacamos los eigenvalores a la matriz de covariancia?
- ¿Que simbolizarían los eigenvectores?
  - Los vectores que describen la manera en la que la covariancia (o la correlación) entre las señales se *comporta*.

Eigenvectores de Covariancia

- Estos estarían MUY relacionados con los vectores de dirección (*direction vectors)* con los cuales modelamos nuestras señales capturadas.

$$\boldsymbol{X_f} = \boldsymbol{A_f} \boldsymbol{S_f}$$

$$\boldsymbol{A_f} = \begin{bmatrix} e^{-i2\pi f T_{1:1}} & e^{-i2\pi f T_{1:2}} & \cdots & e^{-i2\pi f T_{1:D}} \\ e^{-i2\pi f T_{2:1}} & e^{-i2\pi f T_{2:2}} & \cdots & e^{-i2\pi f T_{2:D}} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-i2\pi f T_{M:1}} & e^{-i2\pi f T_{M:2}} & \cdots & e^{-i2\pi f T_{M:D}} \end{bmatrix} \qquad \boldsymbol{S_f} = \begin{bmatrix} s_1(f) \\ s_2(f) \\ \vdots \\ s_D(f) \end{bmatrix}$$

Donde:

$s_d(f)$: es una señal de origen en frecuencia f

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

A_f: matriz que contiene los vectores de dirección *para la frecuencia f*

X_f: señales capturadas *en la frecuencia f*; cada renglón representa un micrófono

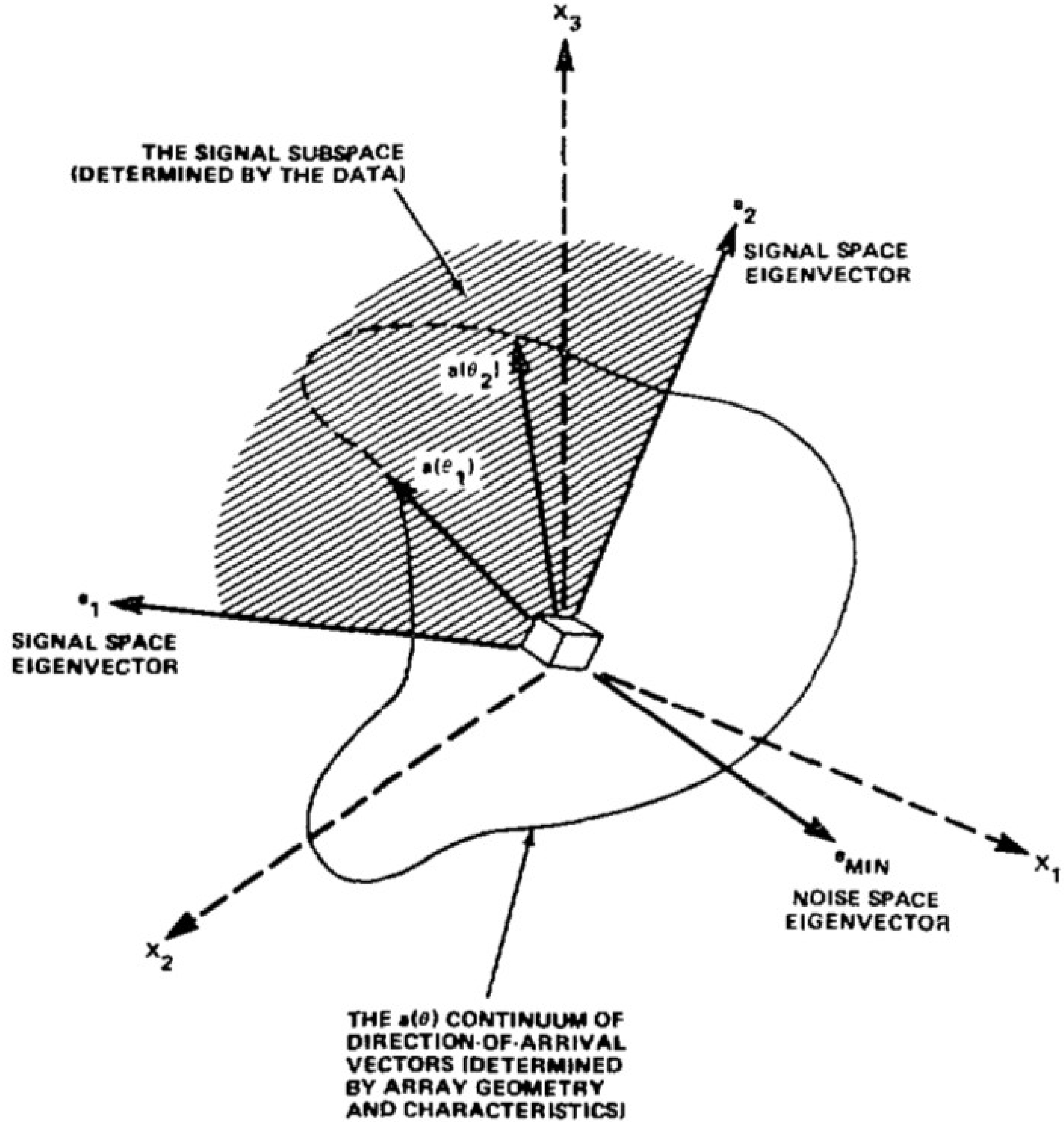
S_f: señales de origen *en la frecuencia f*; cada renglón representa una señal de origen

Direction Vectors

- Recordemos que esto es lo que queremos estimar.
- El direction vector dictamina el desfase temporal que tiene cada señal de origen en cada micrófono.
- Desfase ==> Dirección de Arribo.

Direction Vectors

- Desgraciadamente los eigenvectores NO son los direction vectors.
  - Son sólo una versión de ellos, proyectados de otro espacio matemático.
    - Recordemos que λvm puede tener infinitas combinaciones.



e_1, e_2, e_{MIN} ARE THE EIGENVECTORS OF S CORRESPONDING TO EIGENVALUES $\lambda_1 > \lambda_2 > \lambda_{MIN} > 0$

e_1, e_2 SPAN THE SIGNAL SUBSPACE

$a(\theta_1), a(\theta_2)$ ARE THE INCIDENT SIGNAL MODE VECTORS

¡¿Entonces?! ¡¿Todo esto para nada?!

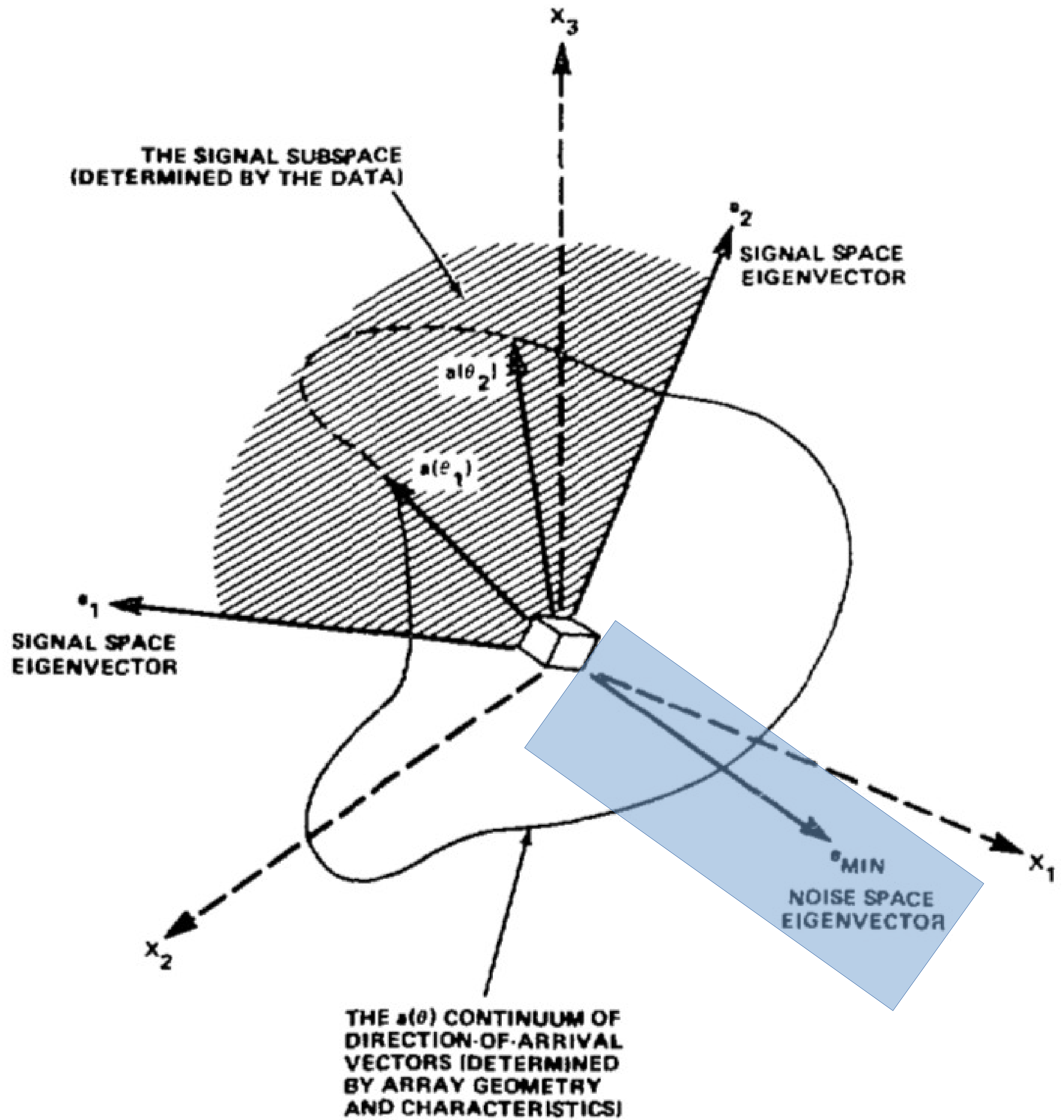
- No necesariamente.

Ejemplo

- Asumamos que tenemos 2 señales de origen y 3 micrófonos.
  - Más micrófonos que señales.
- La matriz de covariancia R será de tamaño 3.
- Lo cual significa que vamos a calcular 3 eigenvectores.

Pero...

- Son sólo 2 señales.
- Realmente nada más 2 de esos eigenvectores están relacionados con los direction vectors.
- El otro vector es... ¿ruido?
  - No necesariamente describe el ruido.
  - Pero está describiendo un subespacio “ruidoso”.



e_1, e_2, e_{MIN} ARE THE EIGENVECTORS OF S CORRESPONDING TO EIGENVALUES $\lambda_1 > \lambda_2 > \lambda_{MIN} > 0$

e_1, e_2 SPAN THE SIGNAL SUBSPACE

$a(\theta_1), a(\theta_2)$ ARE THE INCIDENT SIGNAL MODE VECTORS

La Parte Crítica

- Este eigenvector adicional es ortogonal a los otros eigenvectors.
- Mejor dicho, **este eigenvector “ruidoso” es ortogonal a los direction vectors**.

Entonces

- Podemos probar con varios direction vectors potenciales.
  - Cada uno apuntando a las diferentes direcciones que queramos que tenga el espectro MUSIC.
  - Tantas como queramos.
- Y en cada prueba, sacarle el producto punto con el eigenvector “ruidoso”.
- Si el producto punto es cercano a 0, significa que es ortogonal a éste.
- Lo cual significa que **es un direction vector relacionado con los eigenvectores de las direcciones de la señales.**

Y, ¿como sabemos cuáles son ruidosos y cuáles no?

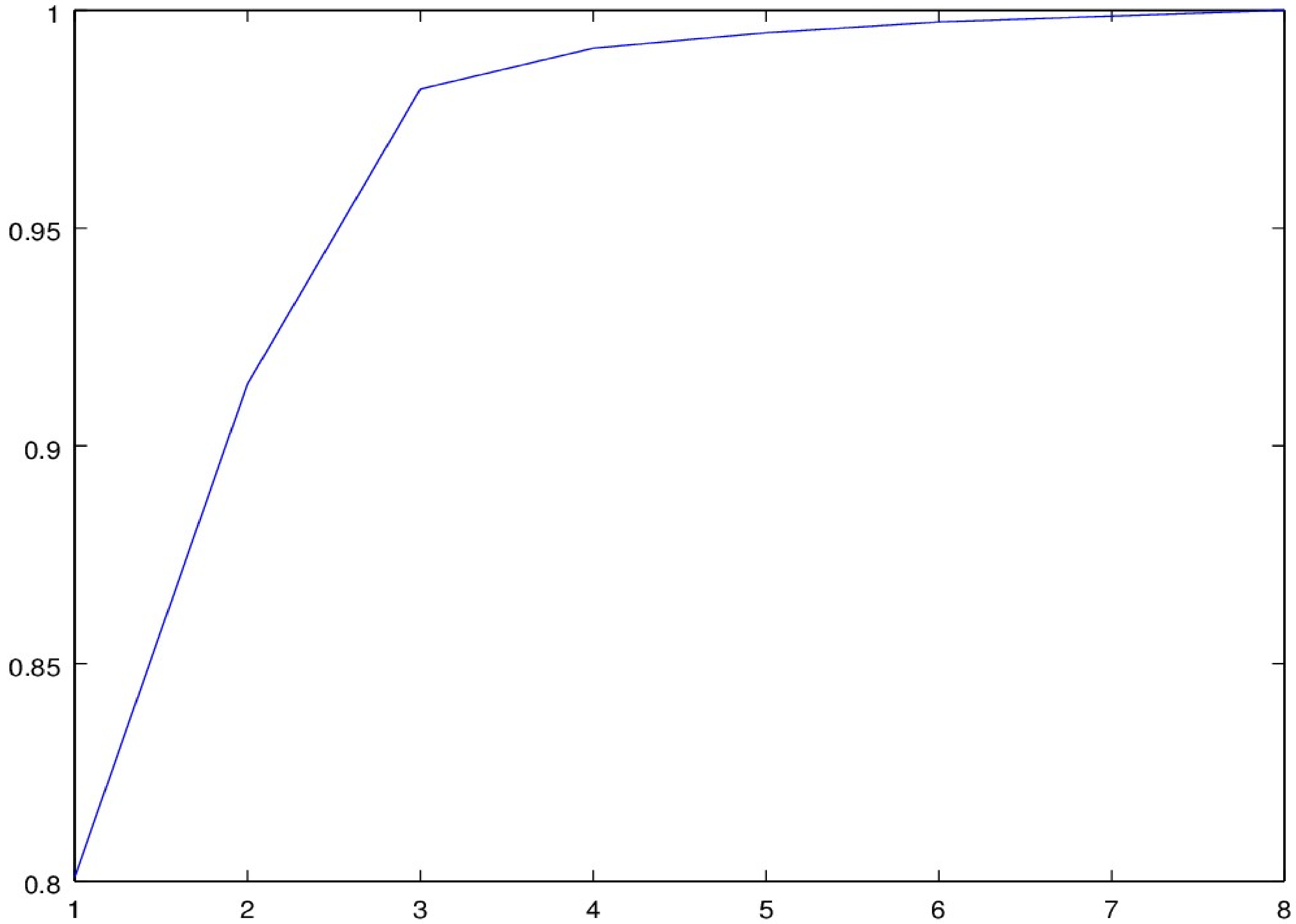
- Por medio de sus eigenvalores.
- Se ordenan los eigenvectores de acuerdo a su eigenvalor.
- Se escogen los que tienen los eigenvalores más grandes.
  - Y los que tienen los eigenvalores más bajos, por ser "ruidosos", tendrán el mismo valor, y uno muy cercano a 0.

Otra forma...

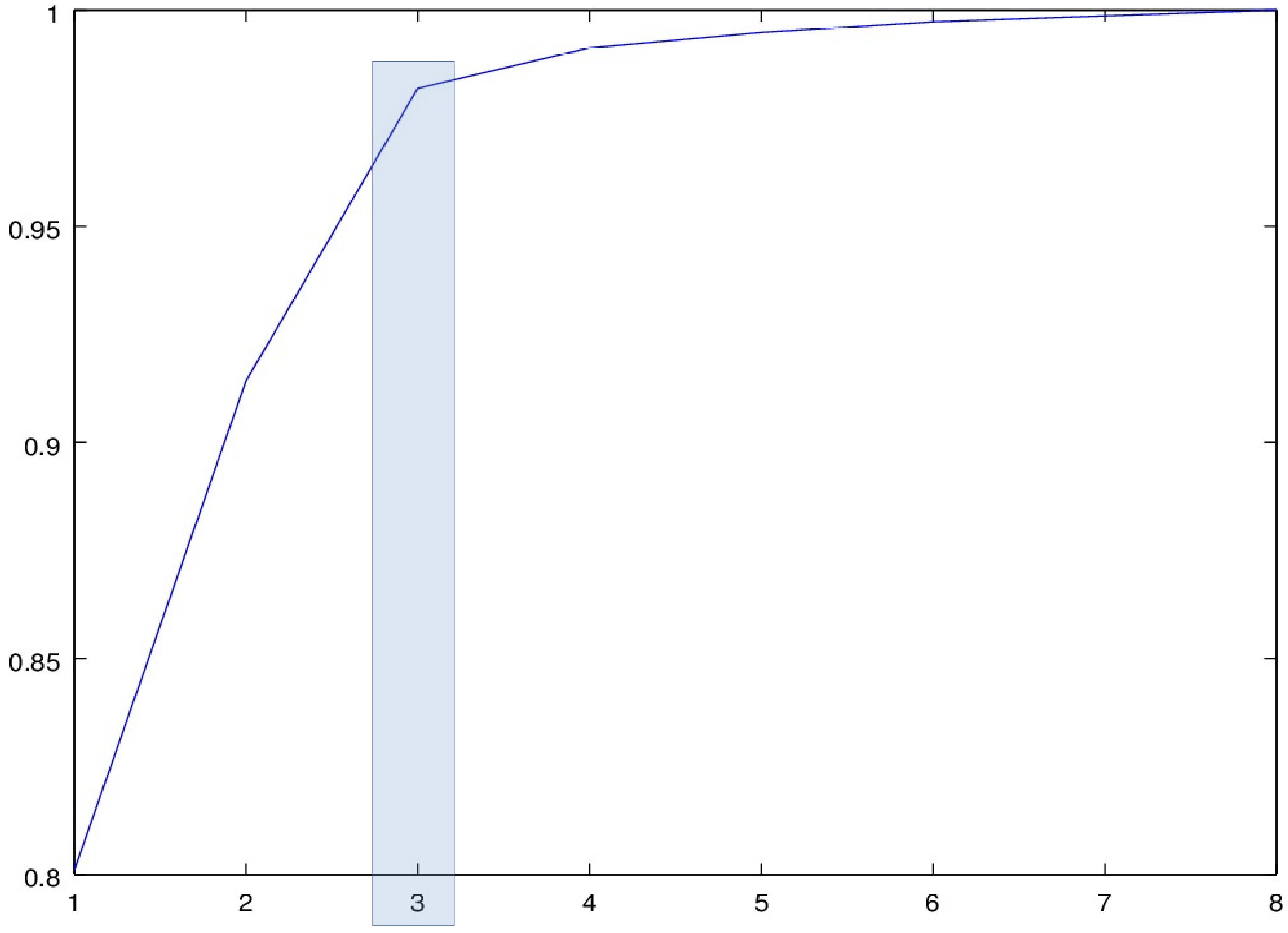
- Recuerden que el eigenvalor es la cantidad de información de la matriz que representa el eigenvector.
- Por lo tanto:
  - Se ordenan los eigenvectores de manera acumulativa, divididos por su suma:
    - Si E es el arreglo de eigenvalores ordenados:

      plot(sumcum(E/sum(E)))
  - Se busca algo cercano a un punto de inflexión.

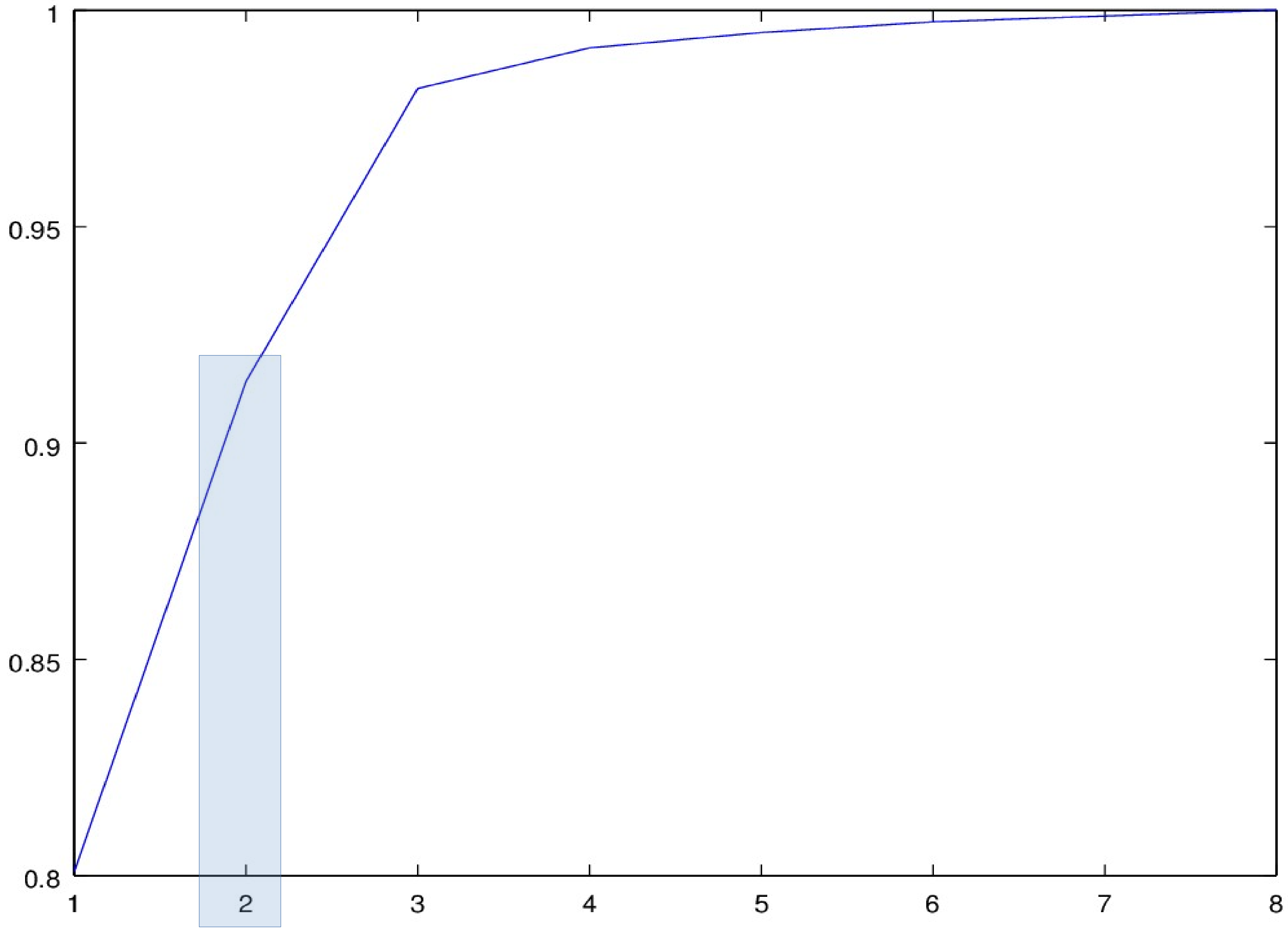
Presentación de Eigenvalores



Presentación de Eigenvalores



Presentación de Eigenvalores



¿En serio?

- Desgraciadamente, la decisión de cuántos eigenvectores utilizar es un tanto abierto.
- Y esto no es el único lugar que vamos a tener que lidiar con esta decisión.

En resumen

$$R = XX^H$$

Se calcula R con datos de una frecuencia.

$$R = V\Lambda V^{-1}$$

$$V = \begin{bmatrix} v_1 & v_2 & \cdots & v_K \end{bmatrix} \qquad \Lambda = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_K \end{bmatrix}$$

Reordenamos a V de acuerdo a los valores en Λ

Decidimos cuáles son los eigenvectores “ruidosos”

Calculamos el producto punto con varios potenciales direction vectors, para producir el espectro de MUSIC.

$$P_{music}(T) = \frac{1}{a_{pot}(T)' * Qn * Qn' * a_{pot}(T)}$$

T: dirección a probar
a_{pot}(T): direction vector a probar ligado a T
Q_n: matriz que contiene los eigenvectors ruidosos
P_{music}(T): valor en el espectro de MUSIC

Notas

- El producto punto está en el denominador para que, cuando dé 0, se obtengan valores grandes y se parezca a los CCVs que hemos estado utilizando.
- El producto punto se calcula así porque la cantidad de eigenvectores es posible que no sea mayor a uno.
  - Así se saca la ortogonalidad de un vector vs. a varios.

Notas

- Claro está, entre más vectores “ruidosos” mayor posibilidad de que la ortogonalidad calculada sea más cercana a la realidad.

Prueba en Octave

- Descarguen:
  - music_complete.m
  - music_multicomplete.m

Ojo

- En estos ejemplos se utilizan señales con frecuencia única.

- Se utiliza toda la señal para hacer la matriz de covariancia, en vez de sólo los valores de su frecuencia.

music_complete

- Crea una señal y la emula entrando en dos micrófonos.
- Calcula el vector MUSIC de -90 a 90 grados, con un incremento de 0.1 grados.
- Presenta dos figuras:
  - Figura 1: la señales capturadas.
  - Figura 2: el espectro MUSIC calculado

music_multicomplete

- Crea dos señales y las emula entrando en tres micrófonos en un arreglo linear.
- Calcula el vector MUSIC de -90 a 90 grados, con un incremento de 0.1 grados.
- Presenta dos figuras:
  - Figura 1: la señales capturadas.
  - Figura 2: el espectro MUSIC calculado

Ruido

- En ambos se puede incrementar el ruido cambiando el valor de:

    noise_w

- Cámbienlo, y corran los scripts varias veces.
  - Recuerden que el ruido se crean con un generador de números al azar.
- ¿El ruido impacta a MUSIC?

Ruido

- En ambos se puede incrementar el ruido cambiando el valor de:

  noise_w
- Cámbienlo, y corran los scripts varias veces.
  - Recuerden que el ruido se crean con un generador de números al azar.
- ¿El ruido impacta a MUSIC?

- No mucho. De vez en cuando algunos picos no aparecen o aparecen picos no esperados, pero es bastante clara la presencia de las señales aún con ruido.

Ojo... de nuevo

- En estos ejemplos se utilizan señales con frecuencia única.
- Generalizar MUSIC a aplicarse a todas las frecuencias es parte de los problemas a vencer si se quieren usar señales complicadas.
  - Como voz.

Recordatorio: Direction Vectors

- Los direction vectors se basan en desfases en el dominio de la frecuencia, y sólo son de UNA frecuencia.

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

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

Donde:

$s_d(f)$: es una señal de origen en frecuencia f

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

A_f: matriz que contiene los vectores de dirección *para la frecuencia f*

X_f: señales capturadas *en la frecuencia f*; cada renglón representa un micrófono

S_f: señales de origen *en la frecuencia f*; cada renglón representa una señal de origen

Resumen de **MUSIC para una Frecuencia**

$$R = XX^H$$

Se calcula R con datos de una frecuencia.

$$R = V\Lambda V^{-1}$$

$$V = \begin{bmatrix} v_1 & v_2 & \cdots & v_K \end{bmatrix} \qquad \Lambda = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_K \end{bmatrix}$$

Reordenamos a V de acuerdo a los valores en Λ

Decidimos cuáles son los eigenvectores “ruidosos”

Calculamos el producto punto con varios potenciales direction vectors, para producir el espectro de MUSIC.

$$P_{music}(T) = \frac{1}{a_{pot}(T)' * Qn * Qn' * a_{pot}(T)}$$

T: dirección a probar
$a_{pot}(T)$: direction vector a probar ligado a T
Q_n: matriz que contiene los eigenvectors ruidosos
$P_{music}(T)$: valor en el espectro de MUSIC

MUSIC de Banda Ancha

- Normalmente se escoge un rango de frecuencias y se lleva a cabo MUSIC para cada una de las frecuencias en ese rango.
- Después se hace un promedio a través de las frecuencias por cada direction vector, “aplastando” los diferentes espectros MUSIC en uno sólo.
  - Se puede utilizar los máximos en vez del promedio.

Problema en Línea

- ¿Creen que se pueda correr MUSIC en línea?
  - La matriz de covariancia se puede crear rápido.
    - Dos for's anidados.
    - Pero se tiene que hacer para cada frecuencia.
  - La eigendescomposición es normalmente lenta.
    - Pero no tanto que sea limitante (Eigen es eficiente).
    - Depende de la computadora.
    - Se puede acelerar utilizando la *descomposición generalizada de valor singular*.
      - Es una generalización de la eigendescomposición pero para matrices rectangulares. También impone restricciones que la hacen eficiente.

Problema en Línea

- Continuación...
  - La búsqueda de direction vectors puede ser lenta si se quiere un espectro MUSIC con alta resolución.
    - Pero esto lo podemos calibrar si fuera necesario.
  - Llevar a cabo MUSIC por cada frecuencia puede elevar mucho el costo computacional.
    - Pero seleccionando las frecuencias apropiadas puede amortiguar dicho costo.

Problema en Línea

- Un problema principal de esta técnica es el requerimiento de recursos de computación elevados para llevar a cabo en línea.
  - Pero “elevados” a finales de los 80's ya no ***debería*** ser un problema ahora.

Técnica:

Basado en Beamforming.

Beamforming

- Este tema está muy relacionado con el tema que veremos después en el curso:
  - Separación de Fuentes por medio Beamforming
- Pero es importante hacerle mención aquí ya que es una forma bastante viable de estimación de dirección de arribo.
  - De hecho, mata a dos pájaros de un tiro.

Beamforming

- Entraré en más detalle después, pero...
- Es una forma de filtrado direccional:
  - Dado una dirección deseada, la técnica reduce las interferencias que provengan de otras direcciones.

¿Dirección Deseada?

- Pero este tema habla de estimar esa dirección.
- Este filtro requiere de esta estimación inicialmente.

- Por eso este tema se ve después, pero, tomando inspiración de MUSIC...

Exploración

- Si tenemos un sistema que pretende obtener la información de audio que proviene de una dirección,
- Podríamos probar diferentes direcciones, y medir la energía del audio que proviene de dicha dirección.
- Si la energía del audio es alta, probablemente ahí hay un señal de origen.

Espectro Beamforming

- De manera similar que con MUSIC.
- Por cada dirección potencial:
  - Se crea un filtro direccional
  - Se mide la energía del audio en esa dirección
- Creando así un espectro beamforming, parecido a los CCVs y espectro MUSIC que hemos estado viendo.

Pero...

- Ya que para poder implementarlo requerimos conocer los conceptos de beamforming, dejaremos los detalles de su implementación para cuando lleguemos al tema de:
  - Separación de Fuentes por medio de Beamforming

Siguiente Clase:

Bases de Separación de Fuentes