La Biblioteca FFTW3
“Fastest Fourier Transform in the West”

Nuevo mejor amigo: FFTW3

- FFT significa “Fast Fourier Transform”.
  - La transformada original suma señales para un coeficiente que se restan en el cálculo de otros coeficientes.
  - La FFT evita estas sumas redundantes, acelerando el cálculo al grado de poderse hacer en tiempo real.
    - Esta “aceleración” fue propuesta por Cooley y Tukey en 1965.
- FFTW3 implementa una versión de esta “aceleración” que es una de las más usadas en aplicaciones de tiempo real.
- Sitio Oficial:

  http://www.fftw.org/fftw3_doc/

Beneficios

- Muy rápida y basada en C.
- Se instala facilmente con apt-get:

  sudo apt-get install libfftw3-dev
- Es código abierto.
  - Repositorio Oficial:

    https://github.com/FFTW/fftw3

Beneficios

- Implementación de números complejos compatible con la de C y la de C++.
  - Para C:

    ```
    #include <complex.h>
    ```

  - Para C++:

    ```
    #include <complex>
    ```

FFTW3 en C

- Se debe incluir **DESPUÉS** de incluir a complex.h

```
#include <complex.h>
#include <fftw3.h>
```

- Para definir números complejos:

```
double complex
```

- Para compilar:

```
gcc jack_fft.c -o jack_fft -ljack -lfftw3
```

Ejemplo Básico C

```
#include <complex.h>
#include <fftw3.h>

//inicializar buffers que usa fftw3
double complex *i_fft, *i_time, *o_fft, *o_time;
i_fft = (double complex *) fftw_malloc(sizeof(double complex) * 1024);
i_time = (double complex *) fftw_malloc(sizeof(double complex) * 1024);
o_fft = (double complex *) fftw_malloc(sizeof(double complex) * 1024);
o_time = (double complex *) fftw_malloc(sizeof(double complex) * 1024);

//configurar a fftw3
i_forward = fftw_plan_dft_1d(1024, i_time, i_fft , FFTW_FORWARD, FFTW_MEASURE);
o_inverse = fftw_plan_dft_1d(1024, o_fft , o_time, FFTW_BACKWARD, FFTW_MEASURE);

… Ya adentro de la función de procesamiento de audio
… llenar a i_time con la señal de tiempo
fftw_execute(i_forward);
… modificar a i_fft para filtrar y copiar el resultado a o_fft
fftw_execute(o_inverse);
… obtener de o_time información filtrada en el dominio del tiempo
```

Ejemplo Completo C FFTW3 ya configurado

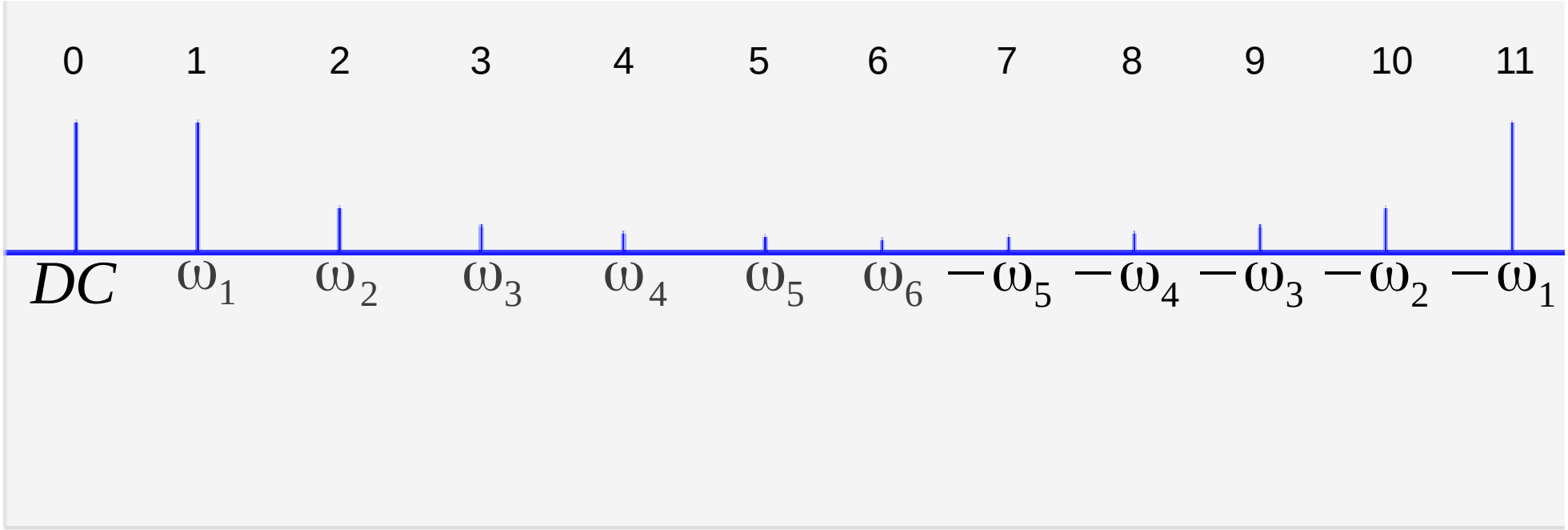
```
// “in” y “out” es nuestra entrada y salida en el tiempo respectivamente
// nframes es el tamaño de la ventana

for(i = 0; i < nframes; i++){
      i_time[i] = in[i];
}
fftw_execute(i_forward);

for(i = 0; i < nframes; i++){
      o_fft[i] = i_fft[i];
}
o_fft[3] = 0;                          //filtrando la tercera frecuencia
o_fft[nframes-3] = 0;                  //espejeando dicha manipulación

fftw_execute(o_inverse);
for(i = 0; i < nframes; i++){
      out[i] = creal(o_time[i])/nframes; //fftw3 requiere normalizar su salida
}
```

Espejeo en C



nframes = 12
i = 3

w_3 → in[3] = in[i]
$-w_3$ → in[9] = in[nframes – 3] = in[nframes - i]

FFTW3 en C++

- Se debe incluir a complex así (sin “.h”):

  #include <complex>

- Para definir números complejos:

  std::complex<double>

- Se requiere utilizar la siguiente función para hacer casting entre los dos tipos de variables complejas:

  reinterpret_cast<fftw_complex>

- Para compilar:

  gcc jack_fft.cpp -o jack_fft -ljack -lfftw3

Ejemplo Básico C++

```
#include <complex>
#include <fftw3.h>

//inicializar buffers que usa fftw3
std::complex<double> *i_fft, *i_time, *o_fft, *o_time;
i_fft = (std::complex<double>*) fftw_malloc(sizeof(std::complex<double>) * 1024);
i_time = (std::complex<double>*) fftw_malloc(sizeof(std::complex<double>) * 1024);
o_fft = (std::complex<double>*) fftw_malloc(sizeof(std::complex<double>) * 1024);
o_time = (std::complex<double>*) fftw_malloc(sizeof(std::complex<double>) * 1024);

//configurar a fftw3
i_forward = fftw_plan_dft_1d(1024, reinterpret_cast<fftw_complex*>(i_time),
reinterpret_cast<fftw_complex*>(i_fft), FFTW_FORWARD, FFTW_MEASURE);
o_inverse = fftw_plan_dft_1d(1024, reinterpret_cast<fftw_complex*>(o_fft),
reinterpret_cast<fftw_complex*>(o_time), FFTW_BACKWARD, FFTW_MEASURE);

… Ya adentro de la función de procesamiento de audio
… llenar a i_time con la señal de tiempo
fftw_execute(i_forward);
… modificar a i_fft para filtrar y copiar el resultado a o_fft
fftw_execute(o_inverse);
… obtener de o_time información filtrada en el dominio del tiempo
```

Ejemplo Completo C++ FFTW3 ya configurado

```
// “in” y “out” es nuestra entrada y salida en el tiempo respectivamente
// nframes es el tamaño de la ventana

for(i = 0; i < nframes; i++){
      i_time[i] = in[i];
}
fftw_execute(i_forward);

for(i = 0; i < nframes; i++){
      o_fft[i] = i_fft[i];
}
o_fft[3] = 0;                          //filtrando la quinta frecuencia
o_fft[nframes-3] = 0;                  //espejeando dicha manipulación

fftw_execute(o_inverse);
for(i = 0; i < nframes; i++){
      out[i] = real(o_time[i])/nframes; //fftw3 requiere normalizar su salida
}
```

Tamaño de Ventana

- Para que la aceleración de FFTw3 funcione, es esencial que el tamaño de ventana sea una potencia exacta de 2:
  - $T = 2^x$, donde x es un número entero.
    - 1024 y 2048 satisfacen esta condición.
    - Además de que es compatible con las necesidades de sobrelape de la función Hann.

Tamaño de Ventana

- Si no cumple con esa condición, se requiere que los datos se “expandan” a potencia exacta de 2 más cercana y mayor, llenando con ceros los que sobran.
- Ejemplo:
  - Tamaño de ventana, *T*: 5 → [1 2 3 4 5]
  - Potencia exacta de 2 más cercana: 8
  - Nueva ventana con *T*:8 → [1 2 3 4 5 0 0 0]
- No le quita exactitud al proceso ni al ventaneo.

Ejemplos Completos

- En la página del curso están ejemplos completos en los que una ventana (o, periodo) de JACK es convertida y regresada al tiempo sin hacerle nada y sin ventaneo.
  - En C: jack_fft.c
  - En C++: jack_fft.cpp
- Compilen y verifiquen que funciona.

ADVERTENCIA

- Este código no está haciendo nada con la información.
  - Sólo se va al dominio de la frecuencia y regreso al dominio al tiempo.

Ejercicio #1

- Filtren todas las frecuencia menores a 500 Hz.
- ¿Cómo sabemos en cual índice de o_fft se ubican dichas frecuencias?
- Recuerden verificar su salida con baudline.

- Si hacemos este filtrado, ¿cómo se escucha? ¿Por qué?

Ejercicio #2

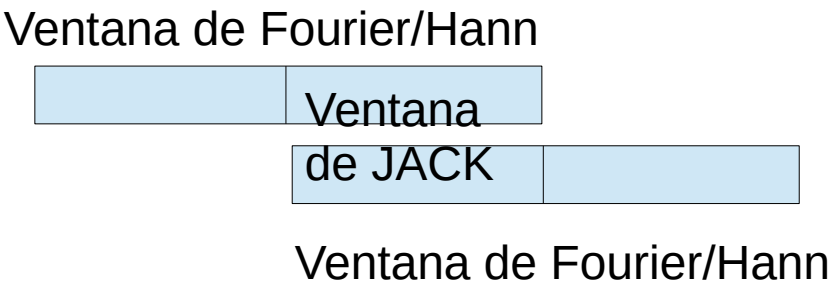
- Hacer un filtro pasa-banda.
- Frecuencia mínima y máxima proporcionados por el usuario.
  - Usen el sistema de argc y argv para obtenerlos.

Sobrelapa-y-Suma

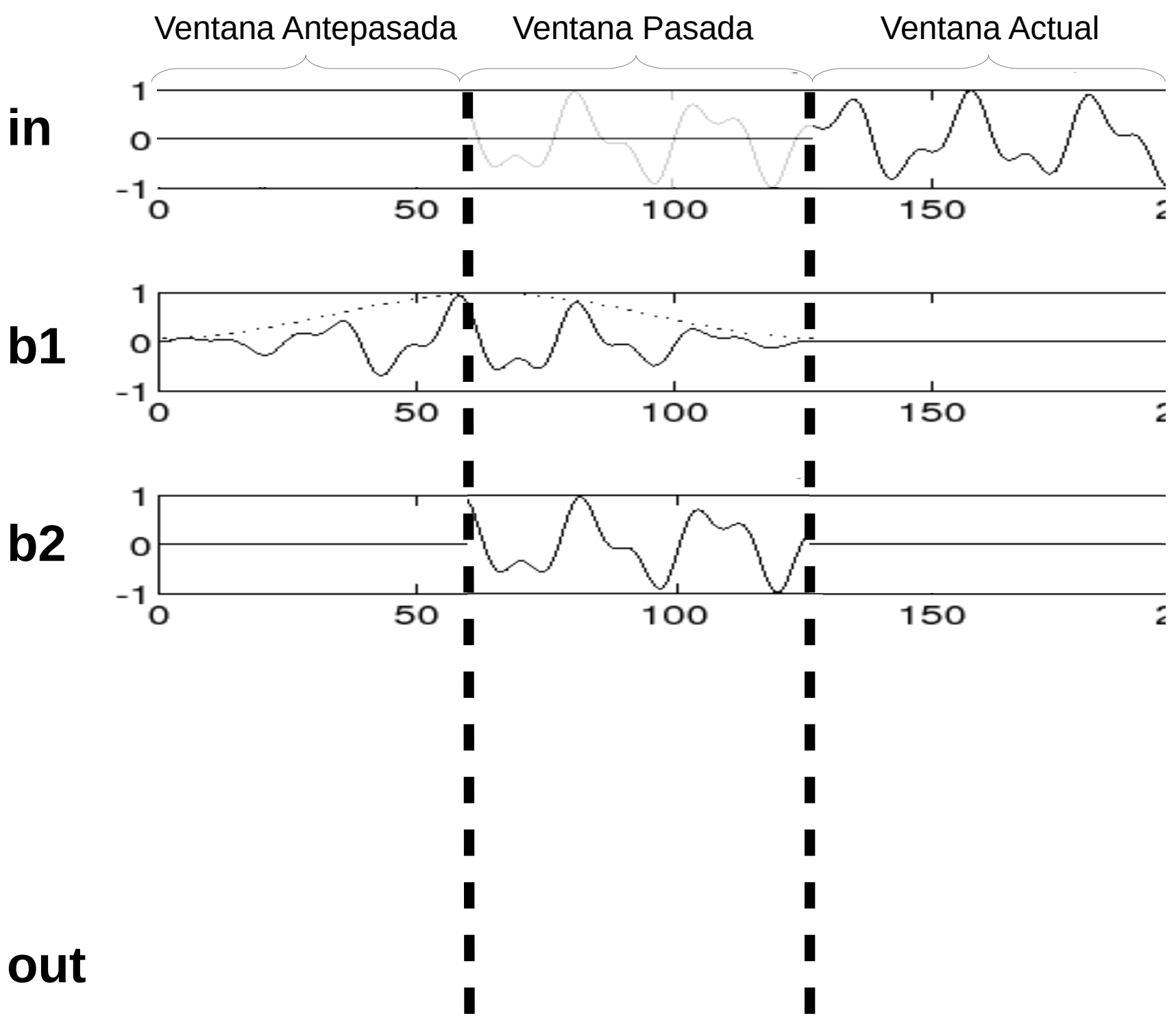
- No lo estamos haciendo todavía.
  - Seguro escucharan “brincos” entre las ventanas.
- FFTW3 nada más proporciona la conversión e inversa de la transformada.
- El ventaneo lo tenemos que hacer nosotros.
- Recuerden utilizar una ventana de tamaño igual a una potencia exacta de 2.

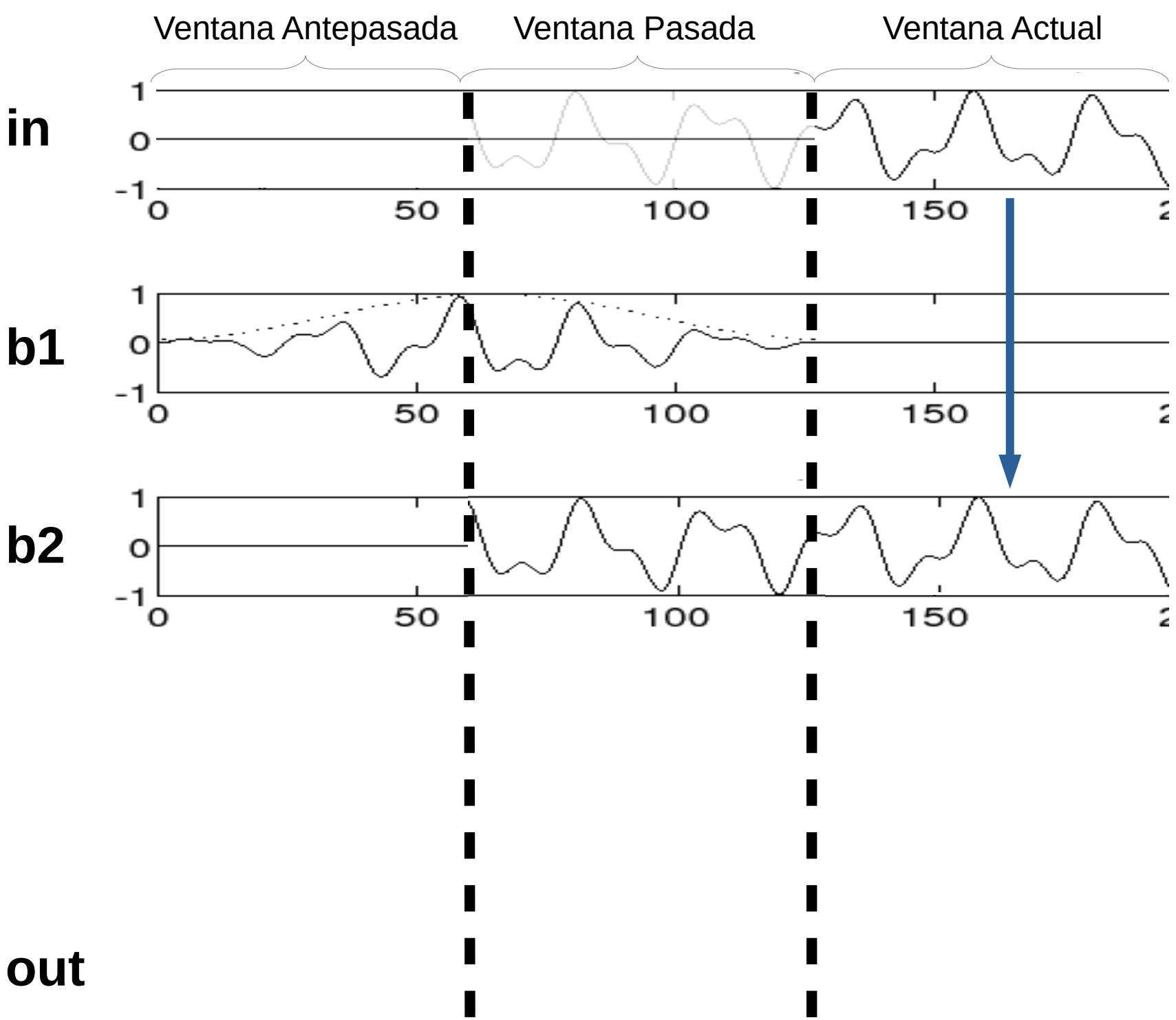
Recomendación Sobrelape-y-Suma

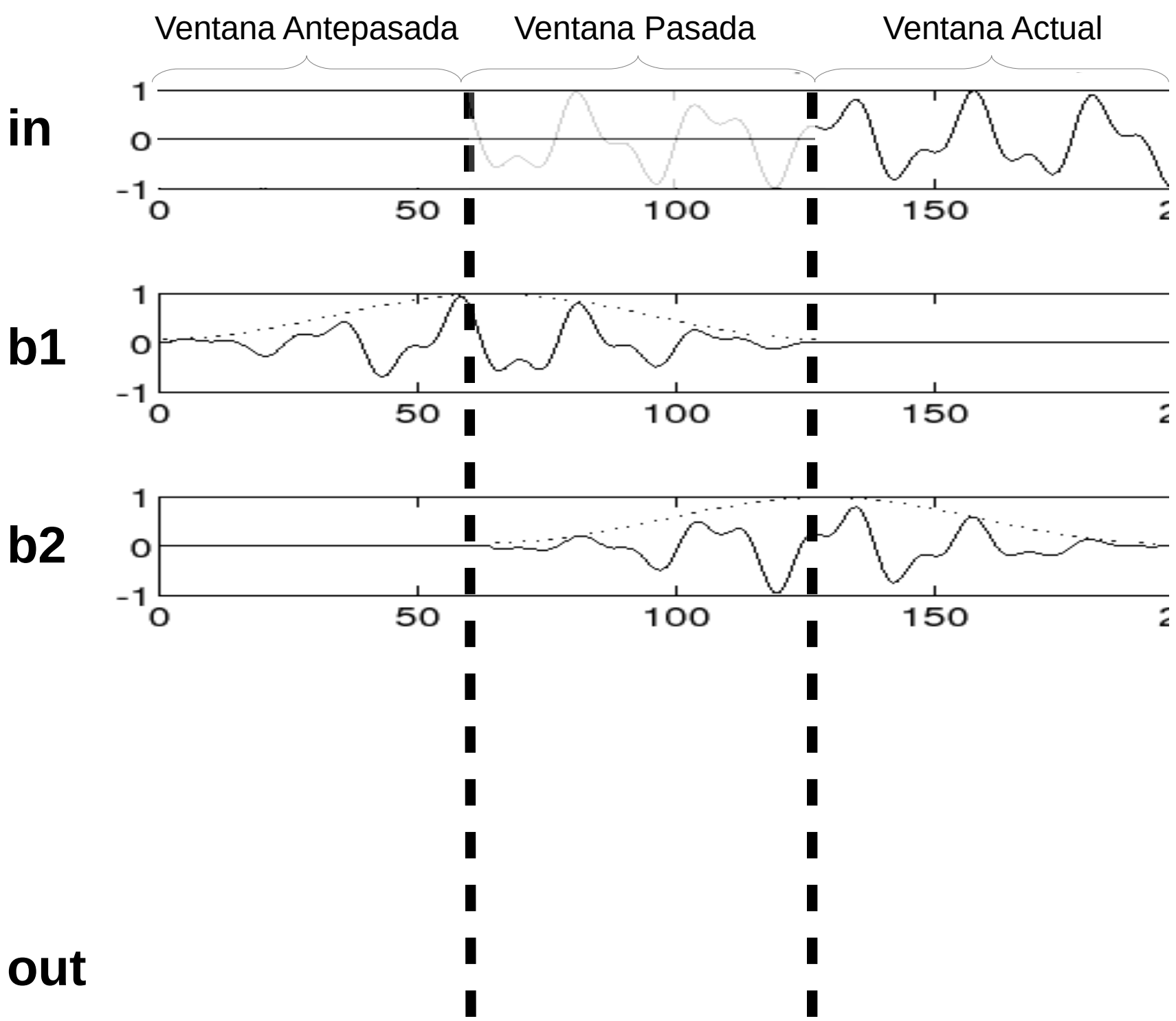
- El tamaño de ventana de JACK, no tiene que ser igual a la ventana que alimentamos a Fourier:

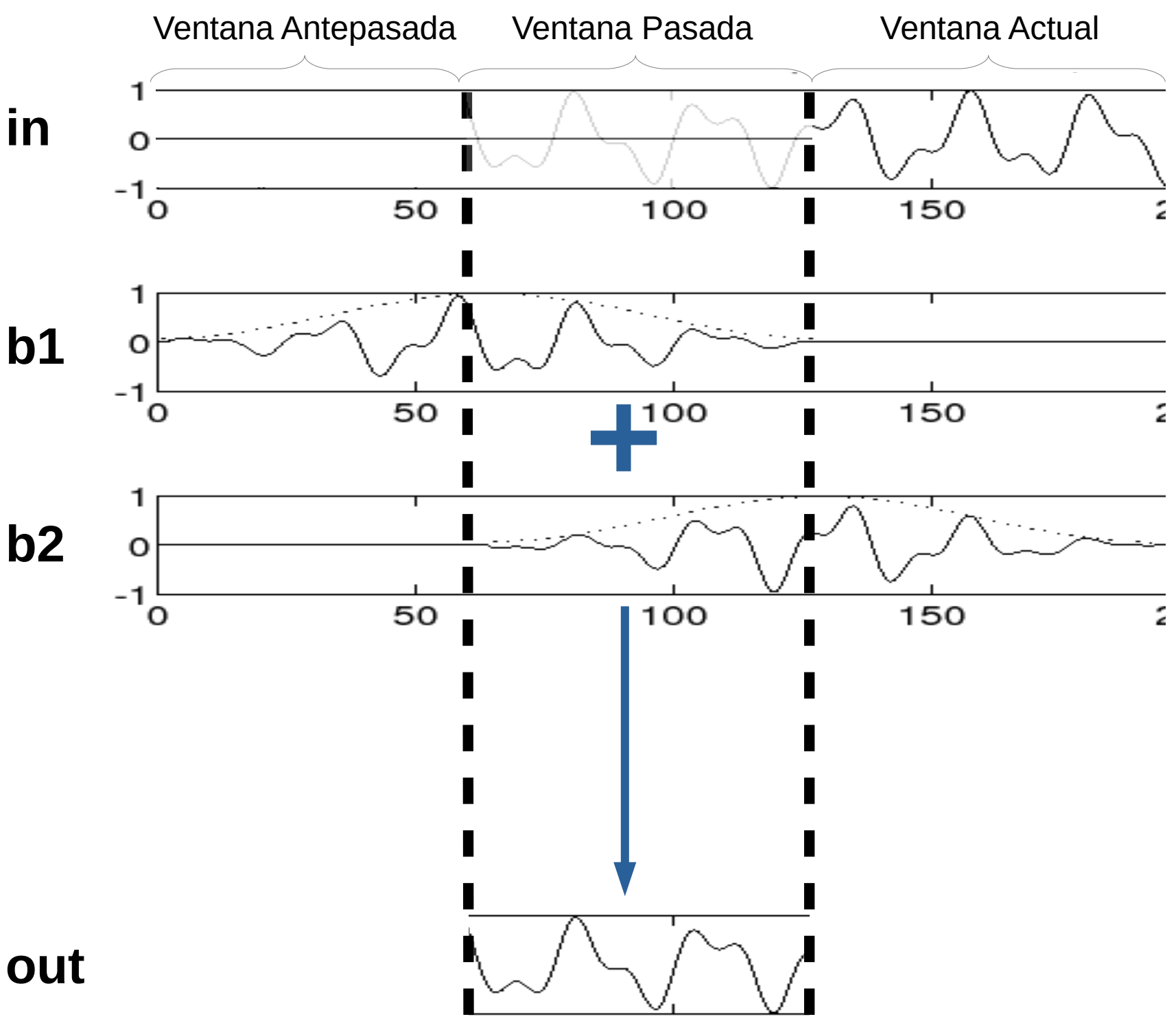


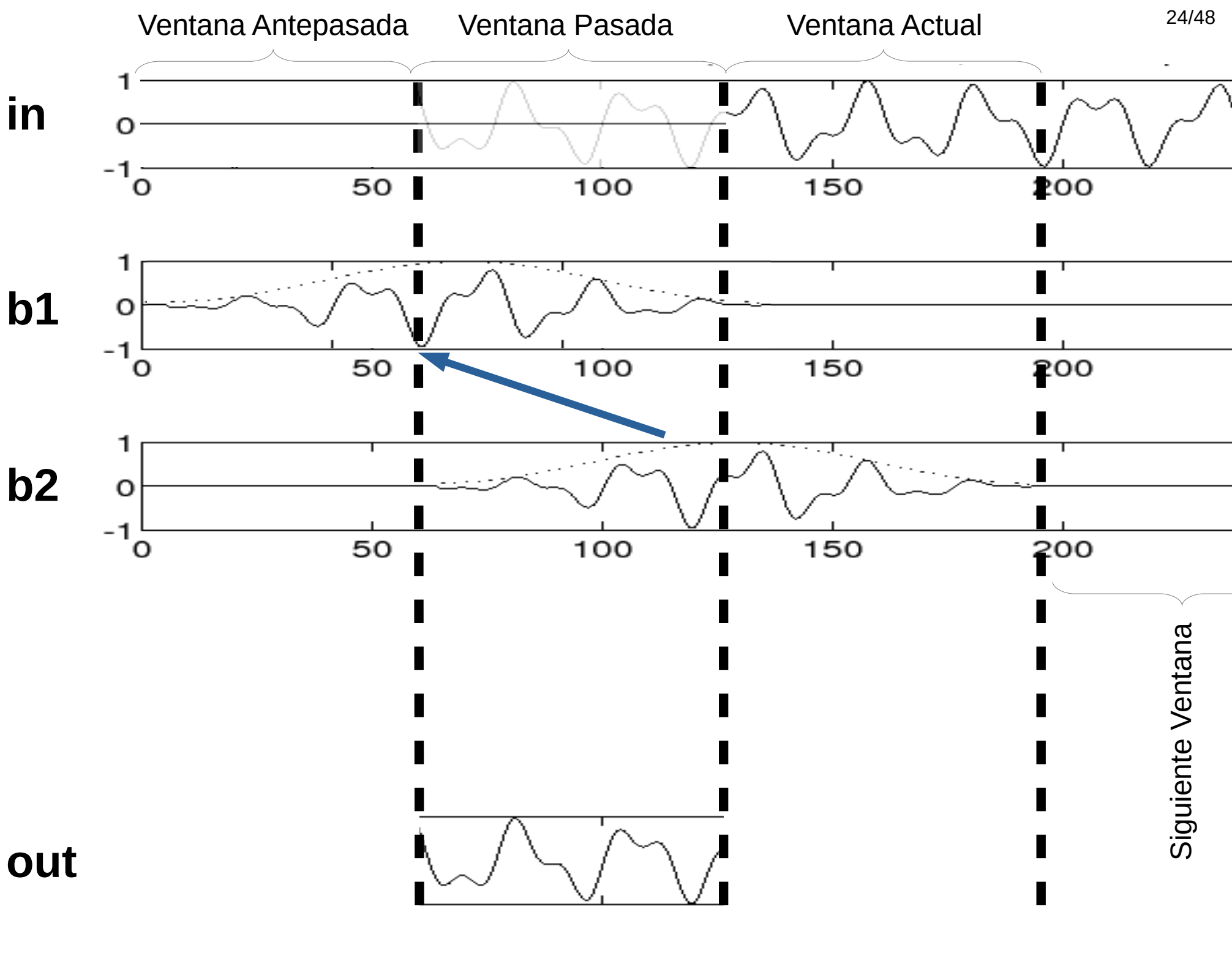
- Utilicen una ventana de Fourier que es el doble de largo que la de JACK.

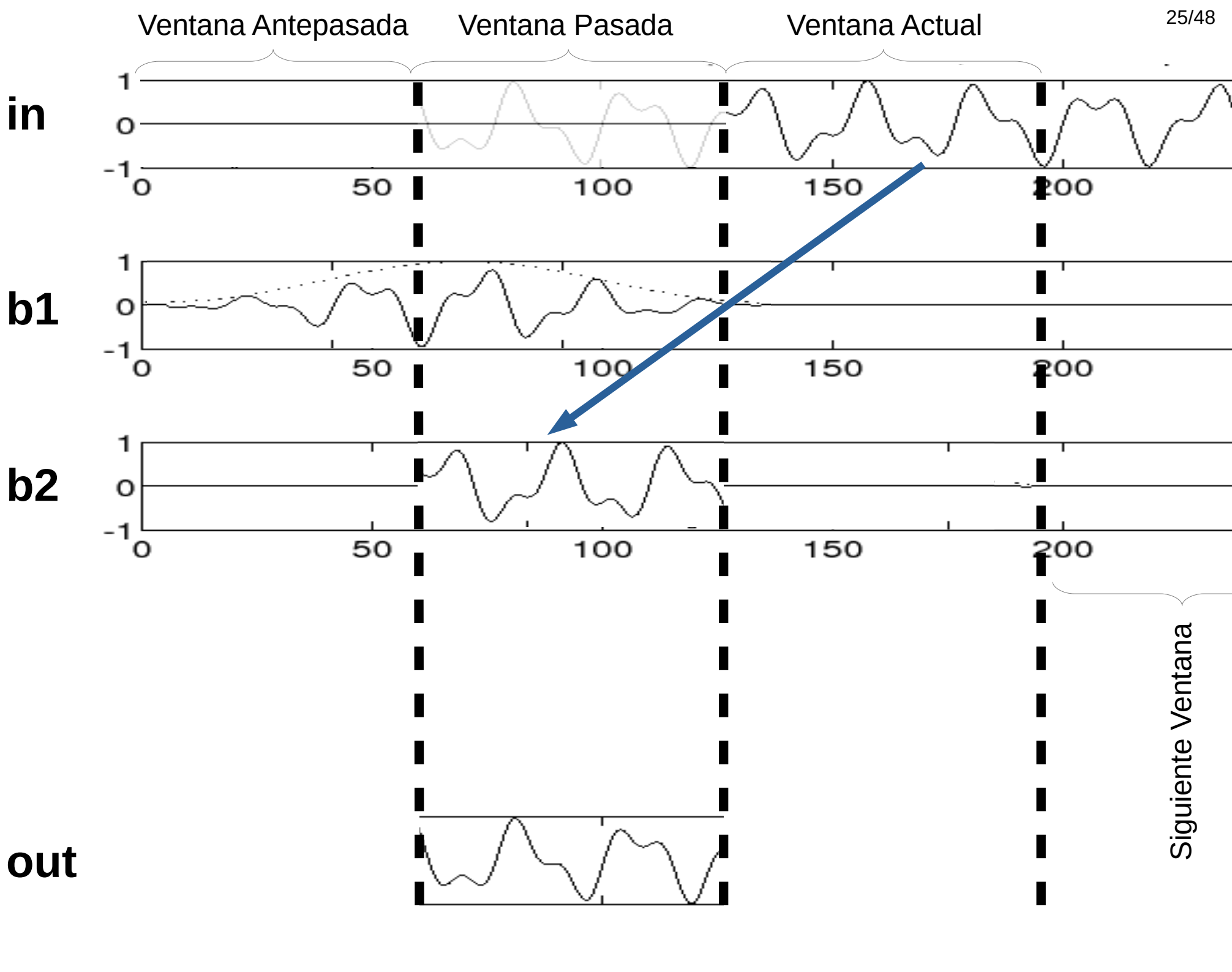

Ventana Antepasada
Ventana Pasada
Ventana Actual
in
b1
b2
out
1
0
-1
0
50
100
150

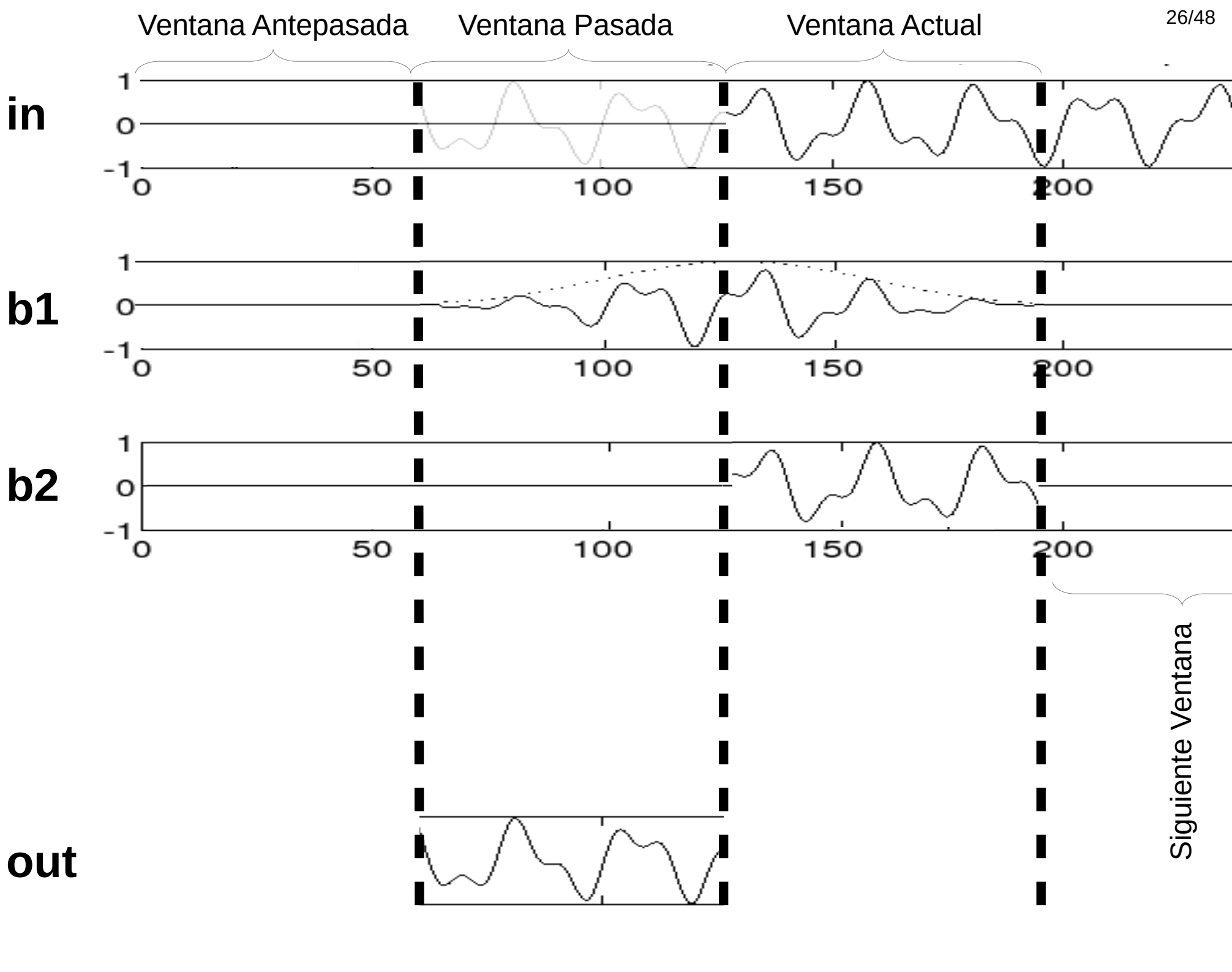

Ventana Antepasada
Ventana Pasada
Ventana Actual
in
b1
b2
out
1
0
-1
0
50
100
150


Ventana Antepasada
Ventana Pasada
Ventana Actual
in
b1
b2
out
1
0
-1
0
50
100
150


Ventana Antepasada
Ventana Pasada
Ventana Actual
in
b1
b2
out
1
0
-1
0
50
100
150


Ventana Antepasada
Ventana Pasada
Ventana Actual
in
b1
b2
out
1
0
-1
0
50
100
150
200
Siguiente Ventana


Ventana Antepasada
Ventana Pasada
Ventana Actual
in
b1
b2
out
1
0
-1
0
50
100
150
200
Siguiente Ventana


Ventana Antepasada
Ventana Pasada
Ventana Actual
in
b1
b2
out
1
0
-1
0
50
100
150
200
Siguiente Ventana

Examen Parcial

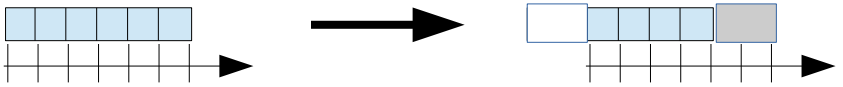
- Hacer un agente de JACK que atrasa (o *desfasa*) la señal de entrada por medio del operador de la exponencial imaginaria, con un argumento en segundos.
- Ejemplo:

  ./programa 0.005

  Atrasa/desfasa la señal por 0.005 segundos.

Desfase por Medio de Exponencial Imaginaria

- Hasta ahorita, hemos desfasado las señales de una manera bastante artificial:
  - Quitando muestras de un lado de la señal, y agregamos ceros.



Desfase por Medio de Exponencial Imaginaria

- Otra forma de desfasar, es por medio de multiplicar la señal transformada Fourier por una exponencial:

$$g(t-T) = F^{-1}\left(G(t)\,e^{-i2\pi\zeta T}\right)$$

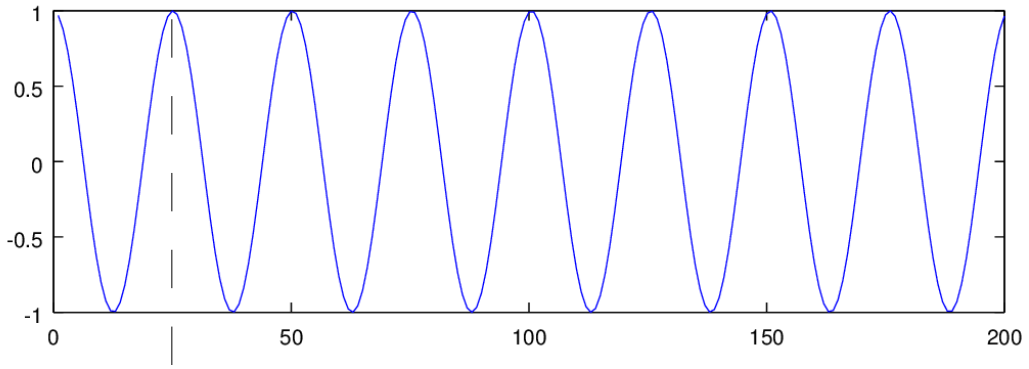
Donde:

g: señal que vamos a desfasar
G: transformada de Fourier de la señal g
T: tiempo en segundos por la que vamos a desfasar la señal
ζ: la frecuencia que vamos a desfasar

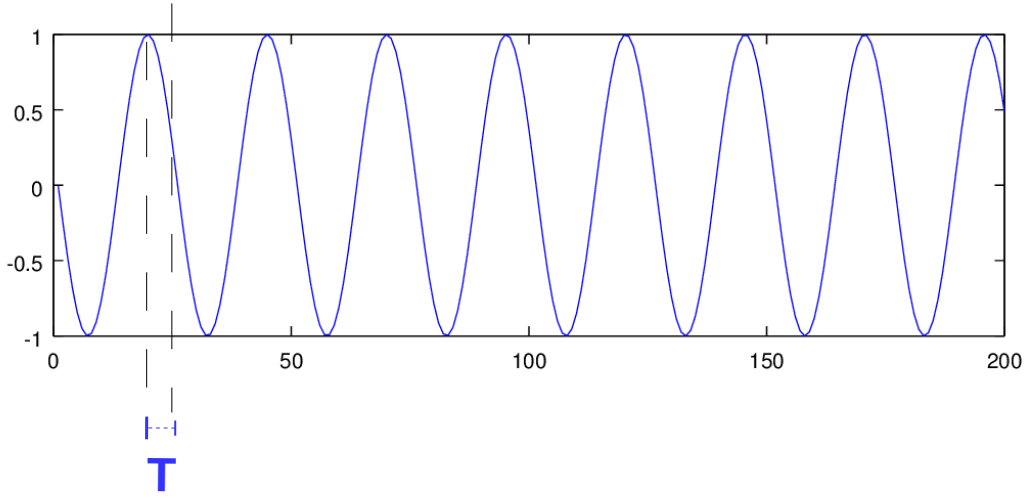
Desfase de Señal en Frecuencia

$$x(t-T) \quad \rightarrow F \rightarrow \quad X(\zeta)e^{-i2\pi\zeta T}$$

x(t)



x(t-T)



¿Frecuencia?

- Estamos desfasando, algo que sucede en el tiempo. ¿Como es que una frecuencia está involucrada?
- Esta *transformación* realmente sólo desfasa los datos de la señal que pertenecen a dicha frecuencia.
- Por lo que si queremos desfasar a TODA la señal, tenemos que aplicar esta transformación a **todas** sus frecuencias.
  - Incluyendo las negativas, con su valor negativo.

Desfase por Medio de Exponencial Imaginaria

- A **todas** las frecuencias se le aplica esto:

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

Donde:

g: señal que vamos a desfasar
G: transformada de Fourier de la señal g
T: tiempo en segundos por la que vamos a desfasar la señal
ζ: la frecuencia que vamos a desfasar

¿Para qué tanto lío?

- Estábamos desfasando las señales de una manera sencilla.
- ¿Para qué complicarnos la vida?

- Varias razones, pero la que nos interesa por ahora es que así podemos desfasar una señal a nivel segundos, no a nivel muestra...

Desfase por Medio de Exponencial Imaginaria

- Es decir...

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

Donde:

g: señal que vamos a desfasar
G: transformada de Fourier de la señal g
T: tiempo en segundos por la que vamos a desfasar la señal
ζ: la frecuencia que vamos a desfasar

Tiempo

- Fourier hace un tipo de extrapolación inter-muestra, lo cual hace esta forma mucho más precisa que hacerlo en el dominio del tiempo.
- Esto es importante cuando estemos haciendo localización y beamforming.

Propiedades de este Desfase

Bidireccionalidad.

- El signo del término T indica el tipo de desfase:
  - Si T > 0: desfase positivo.
  - Si T < 0: desfase negativo.

Espera...

- ¿Desfase negativo? O sea, ¡¿adelantar la señal?!
  - ¿No viola eso algún tipo de ley del mundo físico?

- Al estar haciendo sobrelape, sacamos a la salida la información de ventanas pasadas.
  - Todos los desfases que hagamos son en el pasado.
  - Un desfase negativo sólo desfasa menos que uno positivo.

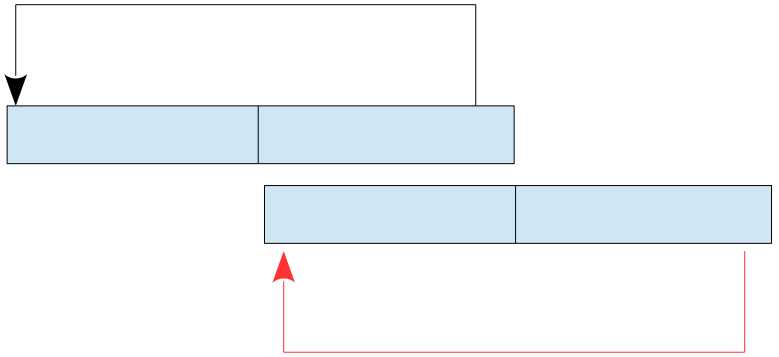
Propiedades de este Desfase

Ciclicidad.

- Por estar aplicando una onda senoidal, el desfase es *cíclico.*
- Dícese: los valores que están al final, que normalmente tacharíamos como “inválidas”, son puestos al principio.

Ciclicidad del Desfase por Frecuencia

- Con un desfase positivo, usando dos ventanas para el proceso de *overlap-and-add* tendríamos:



- Lo contrario sucede con un desfase negativo.
- Esto introduce discontinuidades sólo haciendo sobrelapa-y-suma con Hann Periódica.

Esto lo resolvemos con una técnica que ya conocen...

Solución: WOLA

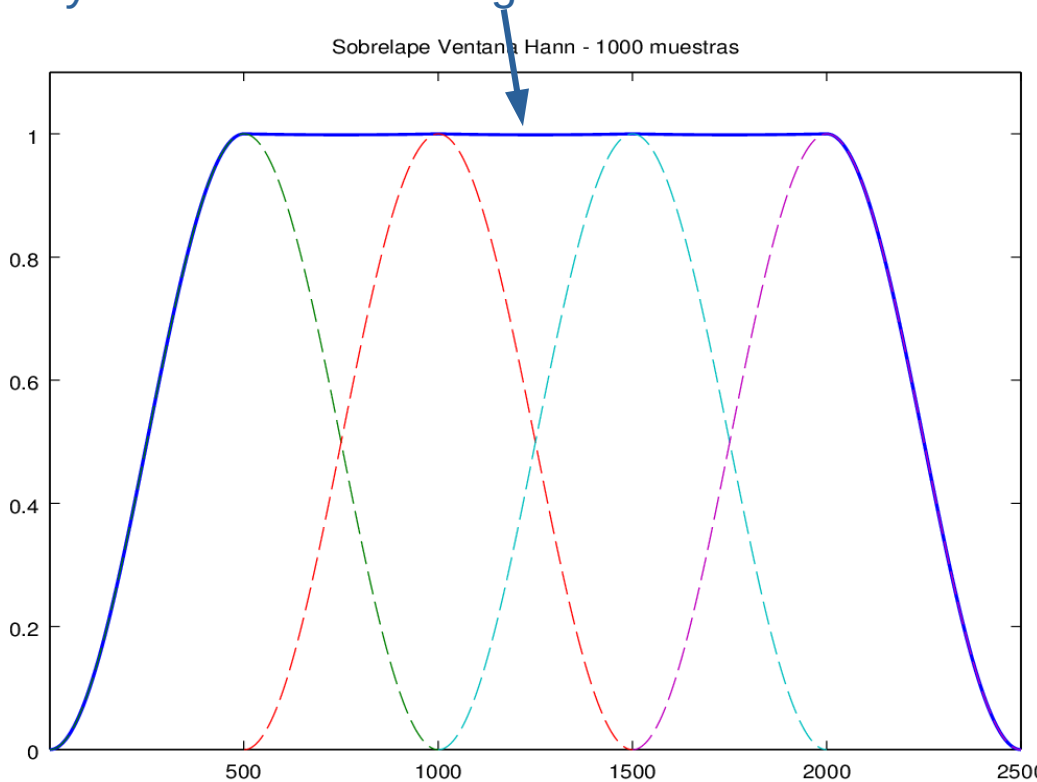
- Así es: WOLA reduce considerablemente el efecto de esta ciclicidad en el resultado final.

- Pero tiene sus límites:

Límite #1

- Ocurre una ondulación a lo largo de varias ventanas, *proporcional a la cantidad de desfase*.

Hay ondulación a lo largo de varias ventanas.



Límite #2

- Sólo maneja desfases menores a media ventana.
- Desfases mayores se “ciclan” a desfases menores.
  - Si la ventana es de 100 ms, un desfase de 75 ms. resulta en uno de 25 ms.

Recomendación

- **Sólo lleven a cabo desfases menores a media ventana de JACK.**
  - La ondulación es casi imperceptible.
  - No se experimenta ciclicidad de desfases.

Pero...

- Si queremos desfases mayores, necesitamos utilizar ventanas de JACK y Fourier más grandes.
  - Lo cual requeriría re-configurar a JACK.
- Aún así...

Media ventana es bastante...

- Si muestrean a 48 kHz, con una ventana de 1024 muestras:
  - Media ventana es alrededor de 0.01 s.
  - La velocidad del sonido es 343 m/s.
  - El sonido recorre 3.43 m. en 0.01 s.

Esto es más que suficiente...

- Si queremos compensar por el desfase entre dos micrófonos, sólo necesitan estar a una distancia menor de 3.43 m.
  - Esto es muy viable para nuestros propósitos.
- Por lo tanto, realmente sólo nos interesan desfases pequeños.
  - Lo cual vamos corroborar en las siguientes sesiones.

Comentarios del Examen Parcial

- Ya que los desfases serán pequeños e imperceptibles por nuestro oído:
  - Deberán revisar con baudline el resultado del desfase.
- Se recomienda tener dos salidas:
  - Una con la señal de “referencia”, en el que se hace el sobrelape pero sin aplicar los operadores de desfase.
  - Otra con la señal desfasada, en el que se hace el sobrelape y aplicando los operadores de desfase.