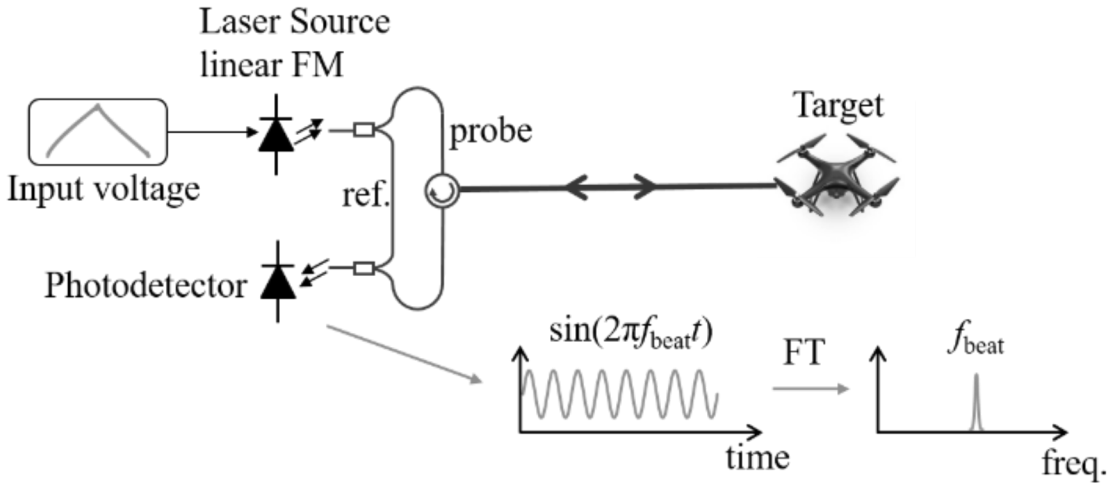
TRABAJO PRÁCTICO N°4

LIDAR FMCW



Armida Abril Paula
Rodriguez Santiago
2023

Índice

Marco teórico

El siguiente trabajo práctico consta de realizar la simulación de una transmisión y recepción de un LIDAR FMCW (Frequency Modulated Continuous Wave). Esta modulación consiste en obtener la frecuencia instantánea de la portadora e incrementarla linealmente entre una frecuencia f_0 hasta una f_1 en un periodo de modulación T_{mod}.

En un radar pulsado se tiene la desventaja de que si se quiere mejorar la resolución de rango (ΔR) o el desvío estándar (σ_R), se debe aumentar el ancho de banda lo cual implica disminuir el ancho del pulso (τ). Esto mismo provoca una disminución de SNR. Es entonces que existe un trade-off entre aumentar la resolución o el desvío y mantener el valor de SNR.

Con FMCW se resuelve dicho inconveniente debido a que los parámetros (ancho de banda y ancho del pulso) se modifican independientemente uno del otro. Por un lado, a través de la modulación se controla el ancho de banda y, por el otro, con el tiempo de modulación se controla el ancho del pulso.

La transmisión y recepción en un LIDAR FMCW es muy similar a la del radar pulsado. En este caso se transmite un láser modulado linealmente en frecuencia, esta señal sufre atenuación, cambio de fase y un delay al pasar por un canal. La atenuación en un LIDAR disminuye con con el cuadrado del inverso de la distancia. Esto se debe a que sólo en la recepción se crea un frente de onda esférico ($1/R^2$). Otros parámetros que afectan la atenuación del canal son la reflectividad de la superficie (ρ) y la apertura óptica del receptor (A_{RX}). El retardo en el canal se debe al tiempo que la señal tarda en ser transmitida y recibida, está determinado por la distancia y la velocidad del medio de transmisión (c).

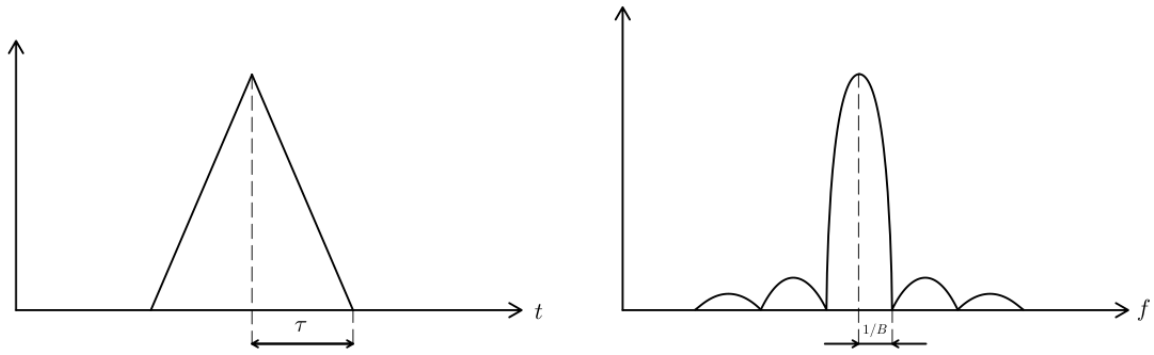
El detector de la señal se lo conoce como *front-end*, el cual consiste en un receptor coherente óptico formado por un conjunto de fotodiodos y una ganancia de detección. Debido a estos componentes eléctricos, se tiene una adición de ruido blanco gaussiano a la salida del mezclador. La potencia de este ruido está asociada con la carga elemental del electrón (q) y la responsividad del diodo (RPD).

A continuación, se encuentra la recepción de la señal transmitida. El receptor utilizado es el *stretch-processor.* El funcionamiento del mismo consiste en conjugar la señal transmitida y multiplicarla con la salida del canal. El resultado de esta operación es un tono cuya frecuencia es proporcional a la distancia del target.

La operación final es la transformada rápida de Fourier de la salida del detector. Aquí se obtiene una señal sinc centrada en la frecuencia del tono obtenido (f_{beat}). El resultado de esta transformada es utilizado para determinar el rango.

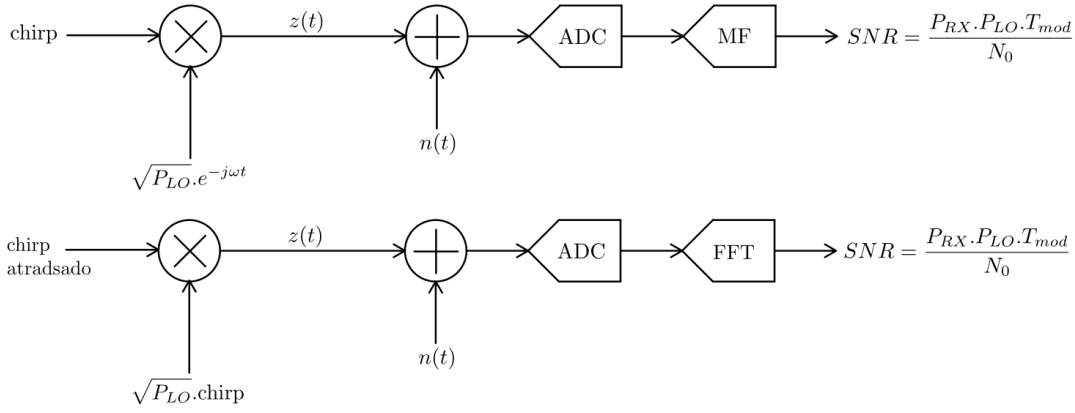
En este caso, la ventaja de utilizar FMCW en comparación al radar pulsado es que la resolución de rango mejora significativamente. Esto se debe a que para obtener la resolución de rango se utiliza el *criterio de Rayleigh*, el cual consiste en saber cual es el primer cruce por cero de la señal a la salida del detector. Con esto, mientras antes ocurra el cruce por cero, mejor será la resolución de rango. Es así que en el radar pulsado, el primer cruce por cero es a una distancia de valor igual a τ respecto al máximo de la señal

recibida y, en FMCW, este valor equivale a $\frac{1}{B}$, lo cual es mucho menor. A continuación se puede observar la diferencia explicada anteriormente.

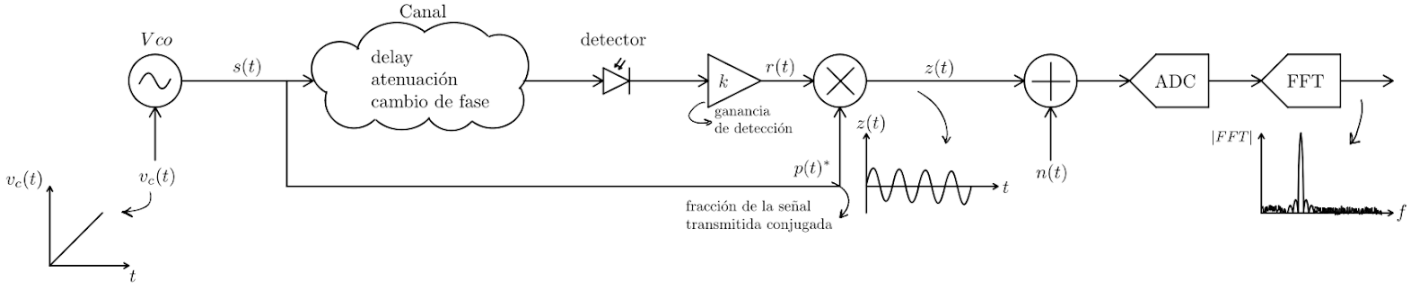


Luego, la característica principal del stretch-processor es que los componentes del receptor pueden tener menor ancho de banda y por lo tanto, se vuelve más económico. Esto se debe a que en el receptor correlador (matched-filter), para poder muestrear la señal recibida, el ADC debe tener una frecuencia de muestreo igual al doble del ancho de banda de la señal transmitida con LIDAR FMCW. Generalmente estos valores son muy elevados, alrededor de las decenas o centenas de GHz. En cambio, el stretch-processor sólo debe muestrear la frecuencia del tono resultante, lo cual equivale a tener un ancho de banda mucho menor que en el caso anterior. Por ende, la frecuencia del ADC en este caso es menor, convirtiéndose así en la opción más económica.

Cabe destacar que utilizando el stretch-processor se sigue manteniendo el mismo valor de SNR que si se usara el matched-filter.



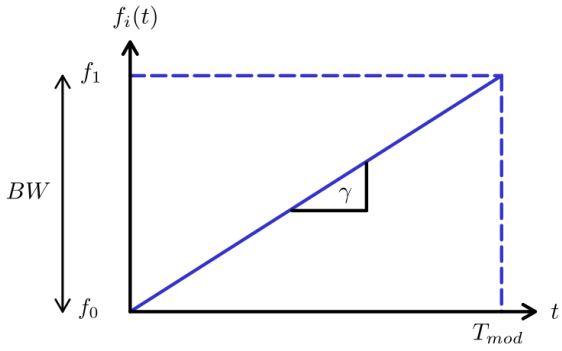
En la siguiente figura se observa un resumen de lo desarrollado.



Ejercicio 1 :

1) **Un transmisor que genere la señal $x(t)$ correspondiente a un chirp de amplitud unitaria, con ancho de banda B y con una duración $T_{mod} \geq T_{meas} + T_{wait}$, donde T_{meas} es el tiempo que se espera capturar señal en el R_x, y T_{wait} es el tiempo mínimo que se debe esperar para cumplir con un rango máximo especificado por el usuario. Considerar una tasa de sobremuestreo NOS , respecto al ancho de banda del chirp.**

Para la generación del chirp es necesario explicar qué es el *Linear frequency modulation.* Este consiste en obtener la frecuencia instantánea de la portadora $f_i(t)$ y hacerla crecer linealmente entre una frecuencia f_o y una f_1 en un periodo de modulación T_{mod}. A continuación se tiene un gráfico ilustrativo.



Es así que la frecuencia instantánea del tono es:

$$f_i(t) = f_o + \frac{f_1 - f_0}{T_{mod}} t$$

Sabiendo que,

$$BW = f_1 - f_o \qquad \text{y} \qquad \gamma = \frac{f_1 - f_o}{T_{mod}} = \frac{BW}{T_{mod}}$$

Entonces,

$$f_i(t) = f_o + \gamma t$$

Luego, la fase instantánea de esta frecuencia es:

$$\theta(t) = \omega_o t + \frac{\pi BW}{T_{mod}} t^2 + \theta_o$$

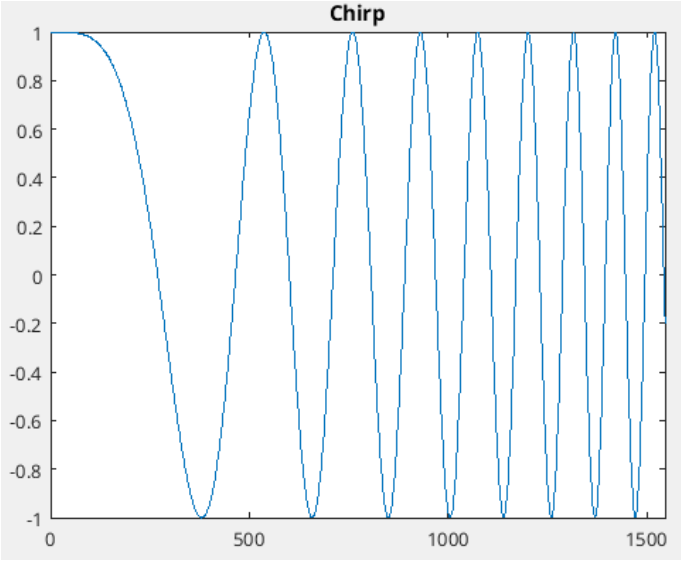
La señal del chirp de amplitud unitaria es:

$$x(t) = e^{j\theta(t)}$$

A continuación se realiza la generación del chirp. Para este se utilizaron los siguientes parámetros:

- $NOS = 4$
- $BW_{chirp} = 2[GHz]$
- $T_{meas} = 2.5[\mu s]$
- $T_{wait} = \frac{2 \cdot max\ range}{c} = 2[\mu s]$
- $T_{mod} = T_{meas} + T_{wait} = 4.5[\mu s]$
- $fs = NOS \cdot BW_{chirp} = 8[GHz]$
- $\gamma = \frac{BW_{chirp}}{T_{mod}}$

La señal del chirp es la que se muestra a continuación.



La imágen anterior se realizó con las siguientes líneas de código.

```
NOS = 4;
fs=NOS*chirp_bw;                          % Frec. de muestreo de Matlab
Ncells = ceil(fs*Tmeas);
fs=Ncells/Tmeas;                          % Frecuencia de muestreo
Ts = 1/fs;                                % Período de muestreo

Lsim = ceil(Tmod*fs);                     % Largo de la sim.
tline = Ts.*(0:Lsim-1)';                  % Linea de tiempo de modulación
insta_freq = chirp_slope.*tline;          % Frecuencia instantánea
insta_phase = 2*pi*cumsum(insta_freq).*Ts;  % Fase instantánea
chirp_tx = exp(1j*insta_phase);           % Chirp unitario
```

2) **Un amplificador con ganancia tal que la potencia transmitida resulte P_{TX}.**

La señal que se quiere transmitir no tiene el mismo valor de amplitud que la señal $x(t)$, esta tiene un valor de $\sqrt{PT_x}$, lo cual corresponde a una señal de potencia PT_x.

$$s(t) = \sqrt{PT_x} \cdot e^{j\theta(t)}$$

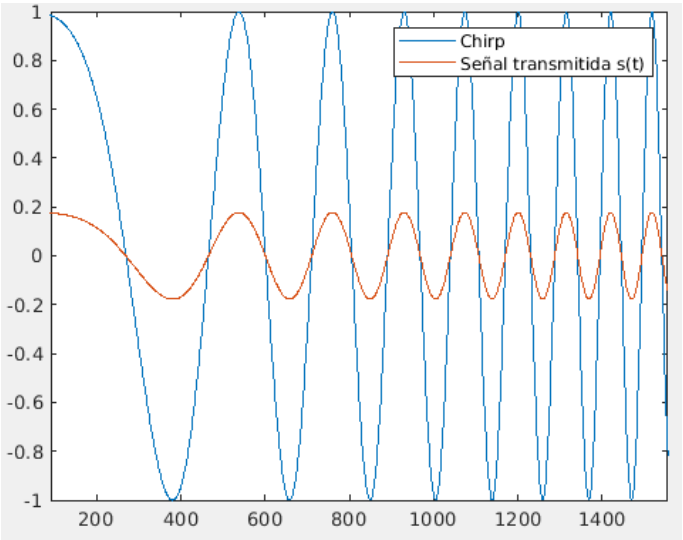
Luego, expandiendo la fórmula anterior:

$$s(t) = \sqrt{PT_x} \cdot e^{j(\omega_o t + \frac{\pi BW}{T_{mod}} t^2 + \theta_o)} = \sqrt{PT_x} \cdot e^{j\omega_o t} \cdot e^{j\frac{\pi BW}{T_{mod}} t^2} \cdot e^{j\theta_o}$$

Se utilizó la siguiente línea de código.

```
s_t = sqrt(PTX).*chirp_tx;   % Señal transmitida
```

A continuación se muestra la comparación entre la señal del chirp y la transmitida con sus diferentes amplitudes.



3) **Realizar un modelo de canal que tenga en cuenta la atenuación** α**, el retardo** ΔT **y el cambio de fase** $\Delta\theta$ **generado por un target.**

Para llevar a cabo la consigna, se aplicaron las siguientes fórmulas para calcular la atenuación, el retardo y el cambio de fase del canal respectivamente:

$$\alpha = \sqrt{\frac{P_{RX}}{P_{TX}}} = \sqrt{\frac{A_{RX} \cdot \rho}{4\pi R^2}} \qquad \Delta T = \frac{2R}{c} \qquad \Delta\theta = e^{j2\pi \frac{c}{\lambda \Delta T}}$$

Teniendo en cuenta la fórmula de atenuación, A_{RX} es la apertura del fotodiodo, ρ es la reflectividad de la superficie y R es el rango. Como se puede observar, la apertura y la reflectividad son directamente proporcionales a la ganancia del canal y el rango es inversamente proporcional. La ganancia del canal disminuye con el cuadrado del rango, esto se debe a que a diferencia de la propagación del radar, el láser se propaga en forma esférica sólo en el camino de vuelta.

Los parámetros utilizados son:

- `PTX = 1e-3*10^(15/10);  % Potencia transmitida [W]`
- `ARX = pi*(2.5e-2/2)^2;  % Apertura de 1 pulgada de diámetro`
- `rho = 0.1;  % Reflectividad`
- `c = 3e8;  % Velocidad de la luz`
- `lambda = 1550e-9;  % long. de onda inicial [m]`
- `omega0 = 2*pi*c/lambda;  % Fase inicial`

Lo primero que se hizo fue calcular la atenuación del canal. Para lo mismo se tuvo en cuenta la fórmula mencionada anteriormente.

```
power_gain = rho*ARX/(4*pi*range.^2);   % Potencia en watts
atten = sqrt(power_gain);               % Atenuación en amplitud
```

Lo siguiente fue obtener el delay. Para esto se calcularon 3 tipos de delay; uno a través de la fórmula de la consigna; otro, relacionado al tiempo discreto que emplea Matlab y el último, el delay en tiempo real (cantidad entera de muestras).

```
tau = 2*range/c;                    % Delay del canal
delay_samples = round(tau*fs);      % Delay discreto
real_tau = delay_samples*Ts;        % Delay en cantidad entera de muestras
```

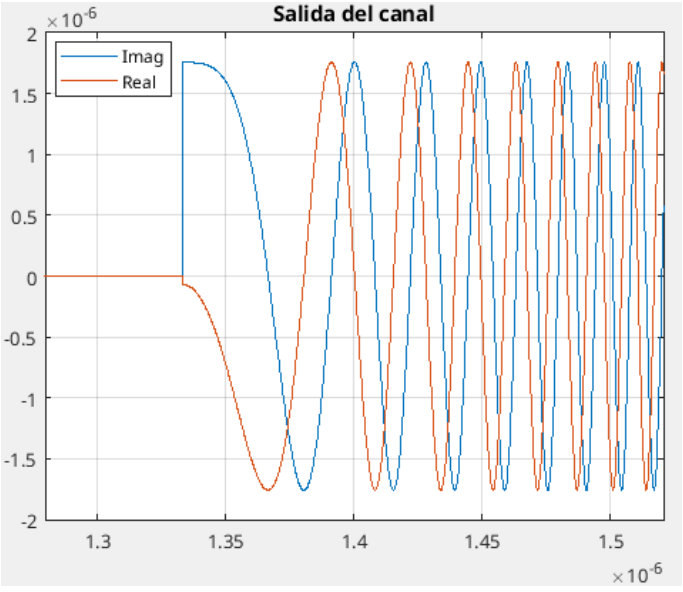
Por último se calculó el cambio de fase con la fórmula de la consigna y lo que se obtuvo es:

```
delta_phase = 2*pi*c/(lambda0*real_tau);        % Cambio de fase
```

El modelo del canal es el siguiente.

```
ch_out = atten.*[zeros(delay_samples,1); s_t(1:end-delay_samples)].*exp(-1j*delta_phase);
```

La señal resultante a la salida del canal es la siguiente.



Se puede observar que tiene un delay ya que la señal inicia a los $1.33[\mu s]$, una atenuación debido a que la amplitud de la señal es de $1.4[\mu]$ y un cambio de fase porque la señal (real) no es un coseno.

4) **Un optical front-end formado por un ICR (Integrated Coherent Receiver). Dado que el modelo del ICR no se ha desarrollado, se propone reemplazarlo por el modelo equivalente de la Fig. 2.**

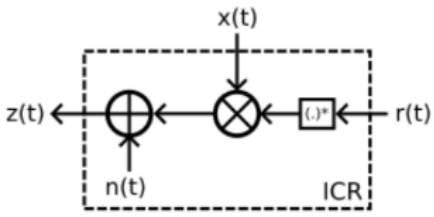


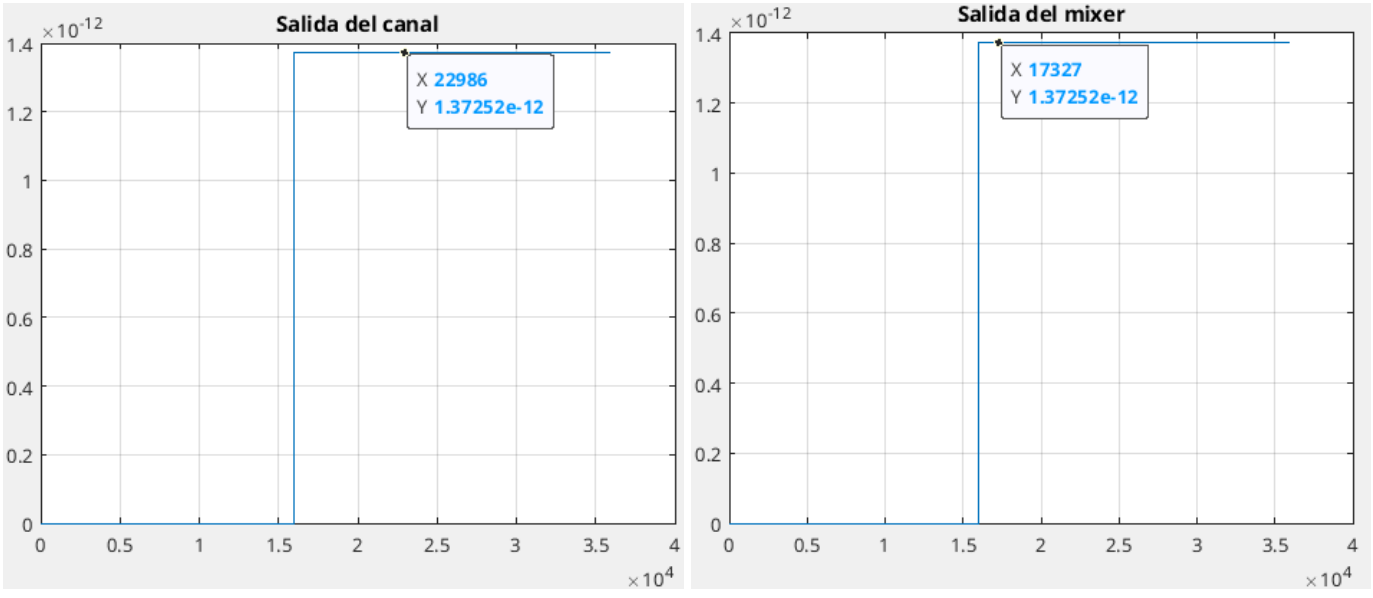
Figura 2: Modelo equivalente del ICR

La señal recibida es conjugada y mezclada con el chirp que se utilizó para la transmisión antes de amplificar, de tal manera que el oscilador local tenga ganancia unitaria. Esencialmente, la señal luego de mezclar debería tener la misma potencia que antes de la mezcla. Luego se agrega un ruido aditivo blanco gaussiano (AWGN) con densidad de potencia $N_o = \frac{q}{R_{PD}}$, donde RPD es la responsividad del fotodiodo (en A/W) y q es la carga del electrón.

El primer paso fue multiplicar el conjugado de la señal transmitida($s(t)$) con la señal recibida ($r(t)$). En la consigna se propone realizarlo al revés, es decir, conjugar la señal recibida y multiplicarla por la señal transmitida, pero esta operación tiene la propiedad conmutativa por lo que se tiene el mismo resultado realizándose de ambas formas. Además, se realiza un conjugado a esta multiplicación con el fin de obtener siempre frecuencias positivas ya que las frecuencias negativas alteran las mediciones en la FFT.

```
mixer_out = conj(ch_out.*conj(chirp_tx));
```

Para comprobar que el oscilador tiene ganancia unitaria, se buscó el valor del máximo de la potencia de la salida del canal y también el máximo de la potencia a la salida del mixer.



Al tener ambos el mismo valor del máximo, no se tiene pérdida de potencia en el mixer y por lo tanto, se tiene ganancia unitaria.

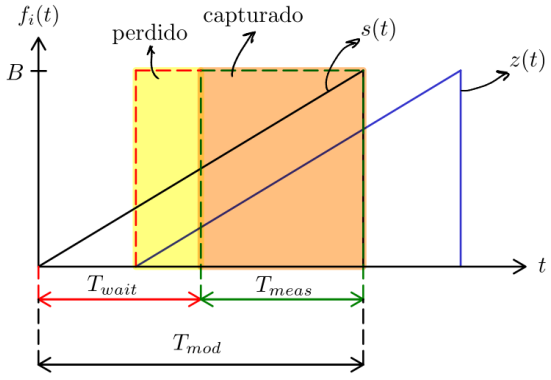
El siguiente paso consiste en adicionar ruido blanco gaussiano a la señal proveniente del mezclador. Este ruido proviene del receptor óptico en donde se encuentran un conjunto de fotodiodos. La potencia del mismo está determinada por la responsitividad del fotodiodo (RPD) y la carga del electrón (q). A continuación se tiene la fórmula.

$$N_o = \frac{q}{RPD} = \frac{1.16\times10^{-19}[C]}{0.1[A/W]} = 1.83 \times 10^{-9}[J]$$

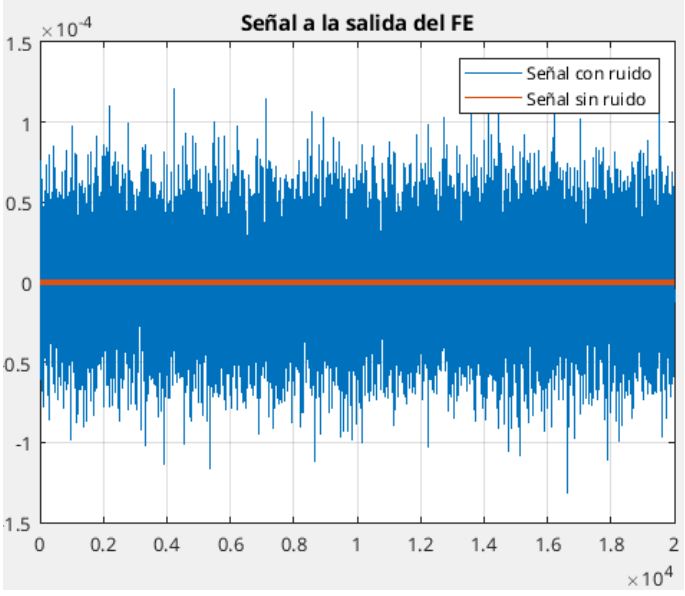
Luego, el ruido es un fenómeno estocástico por lo que variará en cada iteración. Su función es la siguiente.

```
noise = sqrt(noise_power/2).*(randn(size(ch_out))+1j.*randn(size(ch_out)));
fe_output = mixer + noise;      %Salida del detector con ruido
```

Por último, se hace una comparación de la señal antes y después de la adición del ruido. Observar que la señal es ventaneada respecto al tiempo de medición (T_{meas}). Esto se realiza debido a que, para que las señales se mezclen en el receptor, se necesita que estén solapadas y esto ocurre en el tiempo de medición. A continuación un gráfico ilustrativo.



Cuando ocurre el solapamiento durante el tiempo de espera (T_{wait}), se pierde un porcentaje de señal y por lo tanto, también se pierde SNR. Esto ocurre debido a que, durante el tiempo de medición trabaja el ADC, por lo que toda muestra de señal que no se encuentre en el tiempo de medición no será capturada.



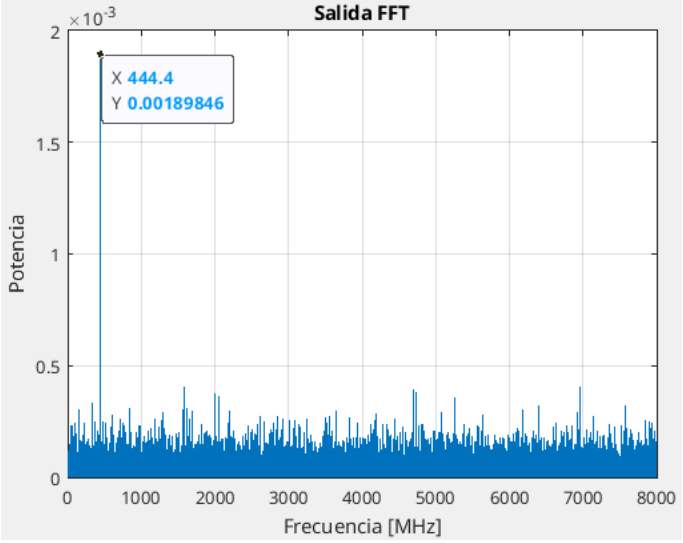
Si bien la amplitud del tono resultante obtenido es mucho menor al piso de ruido, esta podrá ser detectada una vez que la señal sea procesada a través de la FFT.

5) **Un receptor que ejecute el matched-filter para FMCW. Primero, el receptor debe recortar la señal que ingresa, para medir exactamente T_{meas} segundos. Hay que asegurarse que la ventana donde se realiza la captura se corresponda con el momento en que la señal de RX y LO están realmente coincidiendo. Luego, se computa la transformada de Fourier de la señal recortada, usando una FFT de muchos puntos (de tal manera que la FFT sea una buena aproximación de la DTFT). Finalmente, se computa el valor absoluto al cuadrado de la salida de la FFT.**
Para chequear que el simulador funciona, apagar el ruido del sistema y chequear que la salida del detector de envolvente tenga la forma esperada y la posición esperada a la salida de la FFT (e.g. cambiar el rango del target y verificar que el pico a la salida de la FFT se mueva en el eje horizontal). Calibrar las escalas de frecuencia en forma adecuada para visualizar rápidamente el rango real del objetivo.

Para poder llevar a cabo este inciso se utilizó el siguiente código.

```
mixer_v = mixer(1+wait_samples:end);        % Salida del mixer ventaneada
noise_v = noise(1+wait_samples:end);        % Ruido ventaneado
fe_output = mixer_v + noise_v;              % Salida del detector con ruido

Ncells = ceil(fs*Tmeas);                    % Número de bines de la FFT mínimo
FFT_NOS = 16;                               % Sobremuestreo de la FFT
NFFT = FFT_NOS*Ncells;                      % Tamaño de la FFT
fvec = (0:NFFT-1)*(fs/NFFT);                % Vector de frecuencia
y_mf = abs(fft(fe_output, NFFT)).^2;        % Salida del MF/FFT
```

A continuación se tiene la salida de la FFT.



Luego, se busca verificar que la frecuencia del tono obtenida sea la misma que la frecuencia teórica. Para esto se toma el valor máximo de `y_mf` y se lo convierte a valor de frecuencia.

```
[~,fbeat_meas] = max(y_mf);          % Se busca el máximo
fbeat_meas = fbeat_meas*(fs/NFFT);   % Fbeat simulado
```

La frecuencia teórica se calcula con la siguiente fórmula:

$$f_{beat} = \frac{\gamma 2R}{c}$$

Si se corre el código con un experimento de $R = 150\,[m]$ se obtiene la misma frecuencia que la calculadora recientemente.

fbeat	4.4444e+08
fbeat_meas	4.4446e+08

Luego, para poder ver a qué distancia se encuentra el target, sólo es necesario hacer una conversión de frecuencia a rango con la siguiente fórmula.

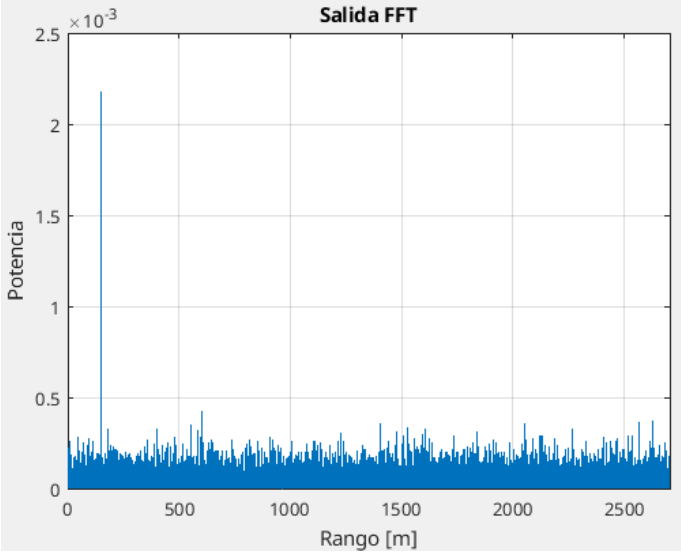
$$R = \frac{f_{beat} \cdot c}{2 \cdot \gamma}$$

Nuevamente se obtuvo el mismo resultado haciendo el cálculo de rango con ambas frecuencias.

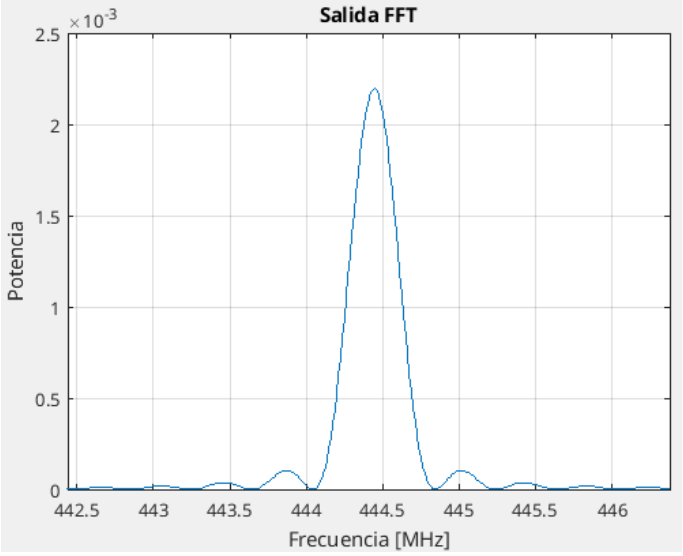
range_meas	150.0061
real_range	150.0000

El código es análogo al utilizado en los cálculos de frecuencia.

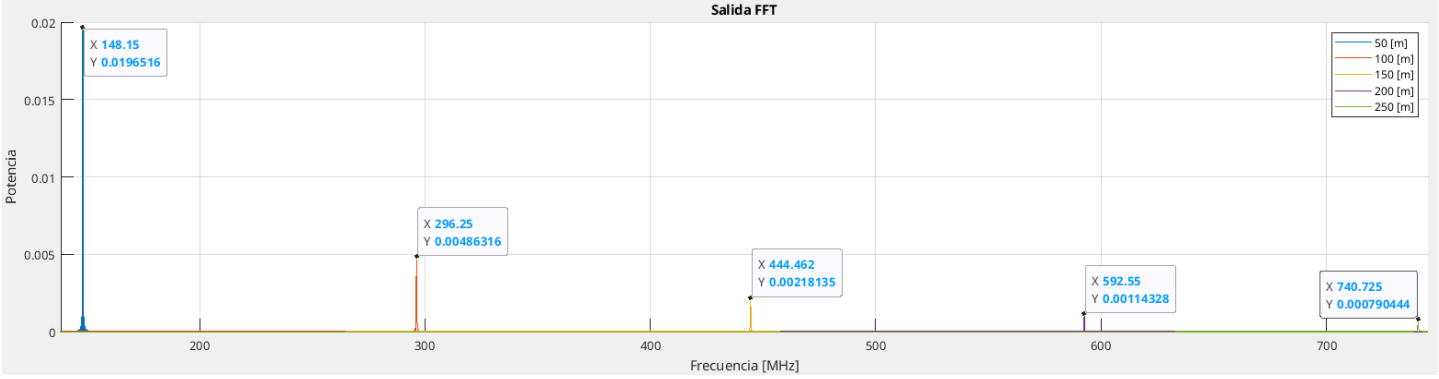
```
max_range_FFT = fs*c/(chirp_slope*2);   % Max. range representado en FFT
rvec = (0:NFFT-1)*(max_range_FFT/NFFT); % Vector de rango
```



Es entonces que, para corroborar que el simulador funciona correctamente, se apaga el ruido del sistema y se observa que la salida del detector de envolvente tenga la forma (un sinc) y la posición esperada a la salida de la FFT. Esto se debe a que a la salida del front-end se tiene un tono ventaneado debido a lo explicado anteriormente del solapamiento de señales. Es entonces, que este tono ventaneado en tiempo equivale en frecuencia a un sinc centrado en la frecuencia del tono.



Por último, se barre la simulación para distintos targets y se comprueba que se mueven en el eje horizontal. Se puede observar que la potencia disminuye con el cuadrado de la distancia.



<u>Ejercicio 2 :</u>

A) **Barrer la posición del target y computar la precisión del LiDAR. Para computar la precisión, realizar N disparos (e.g. N=500) para el mismo target y computar el desvío estándar de la distancia estimada a partir de la señal de salida de la FFT. Para estimar la distancia, asumir por ahora que solo se utiliza la posición del pico máximo a la salida de la FFT. Dibujar la precisión vs la distancia del target.**

Para llevar a cabo este inciso se decidió realizar 500 disparos para cada target y se tomaron 13 targets desde los 50[m] hasta el rango máximo(350[m]) con un paso de 25[m]. Luego se obtuvo el índice del máximo de cada disparo como la distancia estimada y se calculó la desviación estándar de los 500 disparos para cada target.

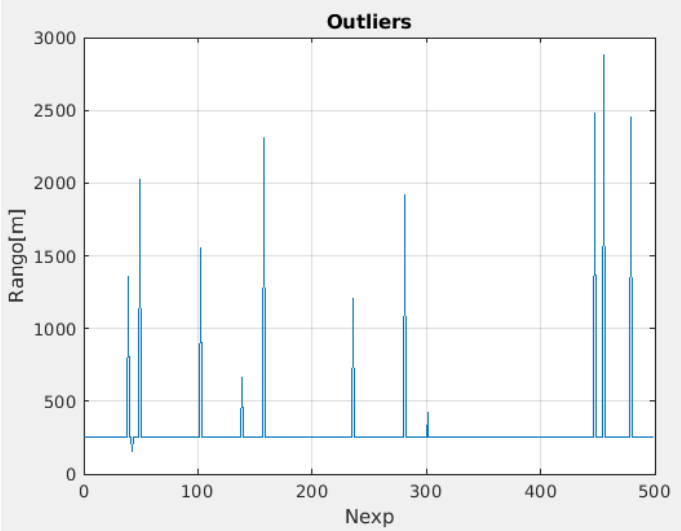
Un problema grave que aparece cuando se calcula la precisión es la presencia de *outliers*. Los outliers son variaciones fuera del rango esperado que se dan en un experimento cuando la SNR es menor a un cierto valor (aprox. 14[dB]), estos deben ser descartados de las muestras tomadas ya que alteran severamente la medición de precisión.

La función utilizada en el código para remover los outliers es la siguiente. Sus parámetros son:

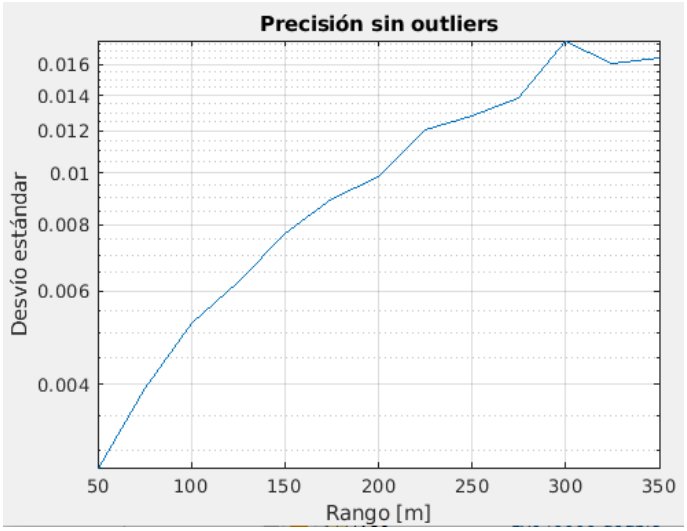
- x: es el vector del cual se quieren remover los outliers.
- mu: el valor medio de las muestras
- thr: threshold o umbral a partir del cual se descartarán las muestras

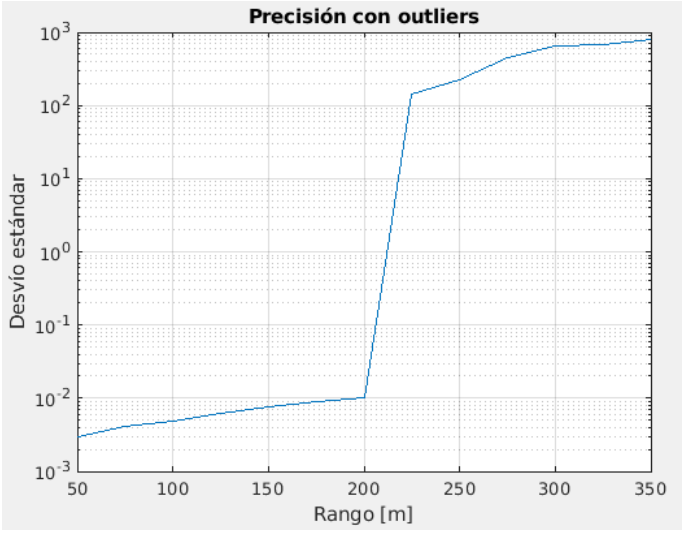
```
function[y] = remove_outliers(x, mu, thr);
y=x; y(x<mu-thr | x>mu+thr)=[];
end
```

Esta función simplemente descarta todo elemento del arreglo que sea mayor a $mu + thr$ ó menor a $mu - thr$. A continuación se observa el resultado de 500 experimentos con la presencia de outliers para un rango de $R = 250[m]$.



Luego, se compara el gráfico de precisión para todos los experimentos en los distintos rangos.





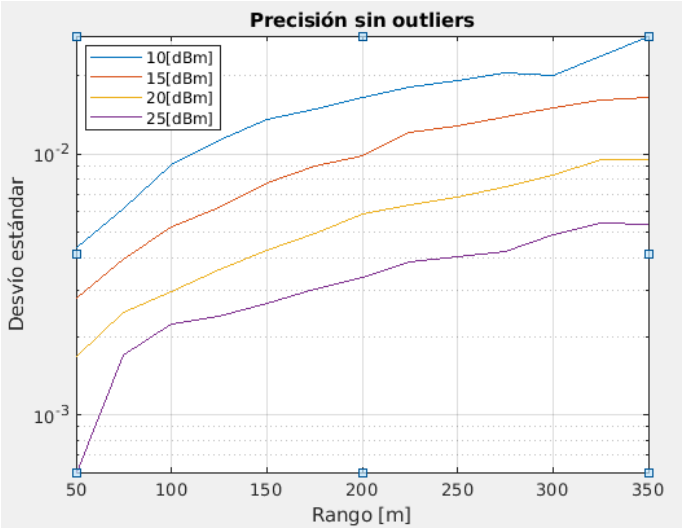
Se puede observar que a mayores distancias, se tienen valores más altos del desvío estándar, eso equivale a un menor valor de precisión. Esto también está relacionado con la SNR. A mayor distancia, se tienen valores más bajos de SNR y por lo tanto, menor precisión.

El código utilizado es el que se puede ver a continuación.

```
Nexp = 500;
f_vals = zeros(Nexp,1);          % Vector de todos los valores de frecuencia obtenidos
r_vals_ol = zeros(Nexp,1);       % Vector de todos los valores de rango obtenidos
r_vals_sol = zeros(Nexp,1);      % Vector de todos los valores de rango obtenidos sin outliers
noise_power = q_elect/RPD*fs;    % Potencia del ruido
wait_samples = ceil(Twait*fs);

for i=1:Nexp
       noise = sqrt(noise_power/2).*(randn(size(ch_out))+1j.*randn(size(ch_out))); % Señal de ruido
       noise_v = noise(1+wait_samples:end);       % Vector de ruido
       fe_output = mixer_v + noise_v;             % Salida del detector con ruido

       y_mf = abs(fft(fe_output, NFFT)).^2;       % FFT

       [~,I] = max(y_mf);                         % Se busca el máximo
       f_vals(i) = I*(fs/NFFT);                   % Fbeat simulado
       r_vals_ol(i) = f_vals(i)*c/(chirp_slope*2); % Rango medido
       r_vals_sol = remove_outliers(r_vals_ol, range, 0.1);

end

range_err = r_vals_ol-real_range;       % Error de rango con outliers
range_err_sol = r_vals_sol-real_range;  % Error de rango sin outliers
range_std(j) = std(range_err);          % Desvío con outliers
range_std_sol(j) = std(range_err_sol);  % Desvío sin outliers
```

B) Barrer la SNR (cambiando la potencia de TX). Superponer todos los resultados y elaborar conclusiones acerca de la variación de la precisión y la SNR (i.e., es lineal?).

Se realizó un gráfico de precisión modificando el valor de la potencia transmitida (P_{TX}) para observar cómo afecta la variación de la SNR a la precisión. Los valores de potencia utilizados son 10, 15, 20 y 25 [dBm] que equivalen a 10, 32, 100 y 316 [mW] respectivamente. Los resultados obtenidos son los siguientes.



Estos resultados se pueden explicar de una mejor manera teniendo en cuenta la fórmula de desvío estándar y la de SNR.

$$\sigma = \sqrt{\frac{\sum_{n=0}^{N} (x-\bar{x})^2}{n-1}} \qquad SNR = \frac{P_{RX} \cdot T_m}{N_o} = \frac{P_{TX} \cdot CH_{gain} \cdot T_m}{N_o}$$

Se puede observar que la SNR es directamente proporcional a la potencia recibida P_{RX}, por lo que, considerando que T_m y N_o no varían, si disminuye P_{RX}, también lo hará la SNR. Teniendo esto en cuenta, en la fórmula del desvío estándar se tiene una sumatoria de $(x - \bar{x})^2$ y cuando la SNR disminuye, el valor de x se vea más afectado por el ruido. Como consecuencia aumenta el valor de la sumatoria y por ende el del desvío estándar.

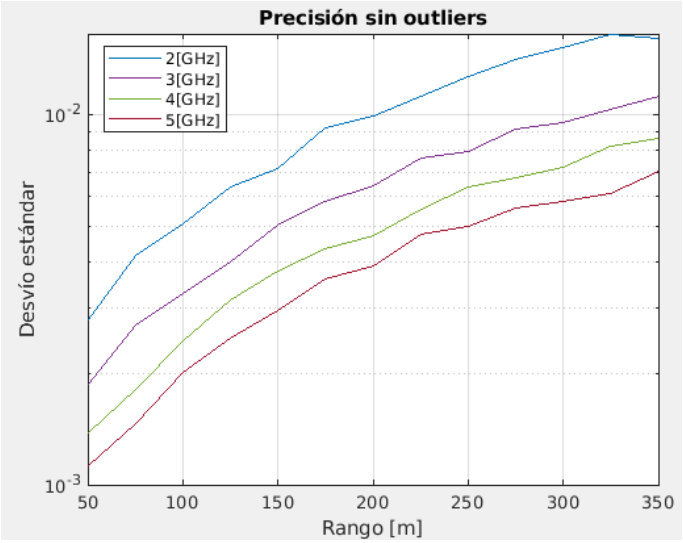
Considerando la siguiente fórmula:

$$\sigma_R = \frac{\Delta R}{\alpha} \cdot \frac{1}{\sqrt{SNR}}$$

Se puede concluir que a medida que aumenta la SNR, el desvío estándar disminuye con la raíz cuadrada de la misma y por lo tanto, la precisión mejora pero no linealmente.

C) Barrer el ancho de banda del chirp (cambiando B desde 2GHz hasta 5GHz). Superponer todos los resultados y elaborar conclusiones acerca de la variación de la precisión vs el ancho de banda (i.e., es lineal?)

Los valores de ancho de banda elegidos para el experimento fueron 2, 3, 4 y 5 GHz. Los resultados se muestran a continuación.



En este caso, se puede observar que la precisión mejora a medida que aumenta el ancho de banda. Si se vuelve a utilizar la fórmula mencionada en el inciso anterior se podrá explicar el por qué ocurre esto.

Por lo tanto,

$$\sigma_R = \frac{\Delta R}{\alpha} \cdot \frac{1}{\sqrt{SNR}}$$

En dicha fórmula se encuentra la resolución de rango (ΔR), esta es la habilidad de un radar de distinguir dos o más targets que están muy próximos en distancia. Su fórmula es la siguiente.

$$\Delta R = \frac{c}{2 \cdot B}$$

Notar que a medida que aumenta el ancho de banda (B), la resolución de rango disminuye. Esto quiere decir que se podrán distinguir targets que se encuentran a distancias cada vez menores. Además, considerando la fórmula del desvío estándar, si la resolución de rango disminuye, también lo hace el desvío y por lo tanto, la precisión aumenta. Se concluye que el ancho de banda y la precisión varían linealmente.

Ejercicio 3:

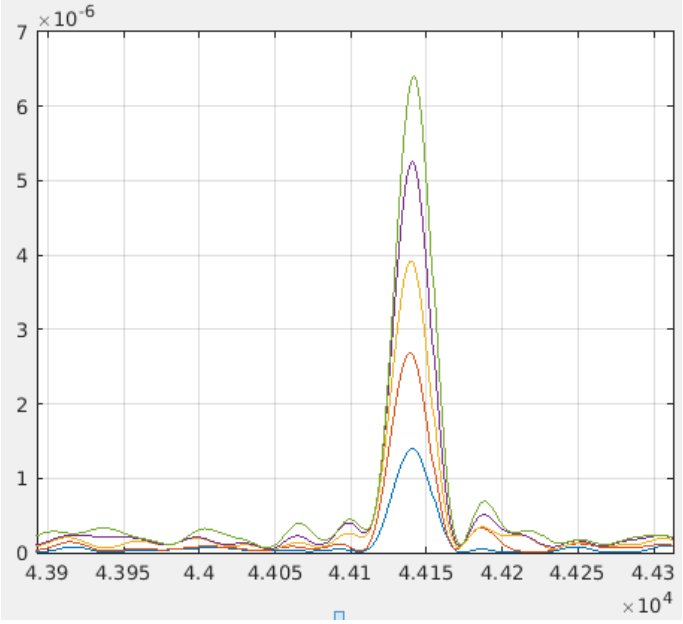
> **Usando el mismo simulador del ejercicio anterior, barrer el rango del target y T_{meas} y computar la SNR teórica. Luego, usando la salida de la FFT, medir la SNR obtenida en simulación para cada caso y superponer la SNR medida a la SNR teórica.**

El cálculo de la relación señal ruido teórica $SNR = \frac{P_{RX} \cdot T_{meas}}{\frac{q}{RPD}}$ siendo la potencia recibida $P_{Rx} = P_{TX} * CH_{gain}$ que es la potencia transmitida por la ganancia del canal.

La SNR teórica se calcula utilizando la siguiente línea de código.

```
prx_theo = PTX*power_gain;                  % Valor de la potencia recibida (teórico)
theo_snr(j) = prx_theo*Tmeas/(q_elect/RPD); % SNR Teórica
theo_snr_dB(j) = 10*log10(theo_snr(j));     % SNR Teórica [dB]
```

Luego, para la SNR computada primero se debe crear una variable (`y_mf_accum`) la cual tomará los valores de la salida del match-filter y los irá acumulando para todos los experimentos en cada rango. Una aclaración importante es que el sinc a la salida del match-filter es determinístico, por lo que en cada disparo este cambiará muy poco (debido a la desviación causada por el ruido). En cambio, el ruido es una señal estocástica con media cero y en cada experimento se irá promediando, por lo tanto, su amplitud tenderá a ser cero mientras que su potencia va a tender a un valor fijo. A continuación se tiene un esquema referido a lo mencionado anteriormente.

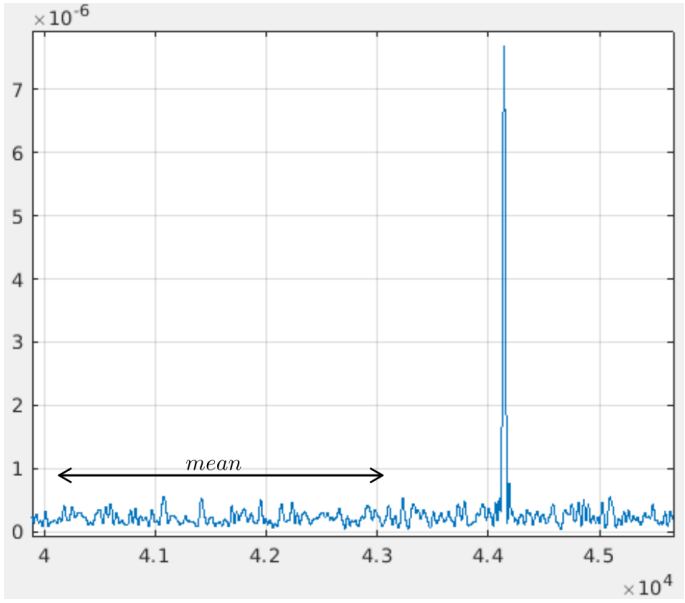


En la imagen anterior, el color azul corresponde a la primera iteración, el color rojo a la segunda y continúa hasta que la potencia del ruido y del pulso correspondan con los valores de potencia asignados en el modelo.

El código utilizado es el siguiente.

```
if i==1
    y_mf_accum = y_mf/Nexp;
else
    y_mf_accum = y_mf_accum+y_mf/Nexp;
end
```

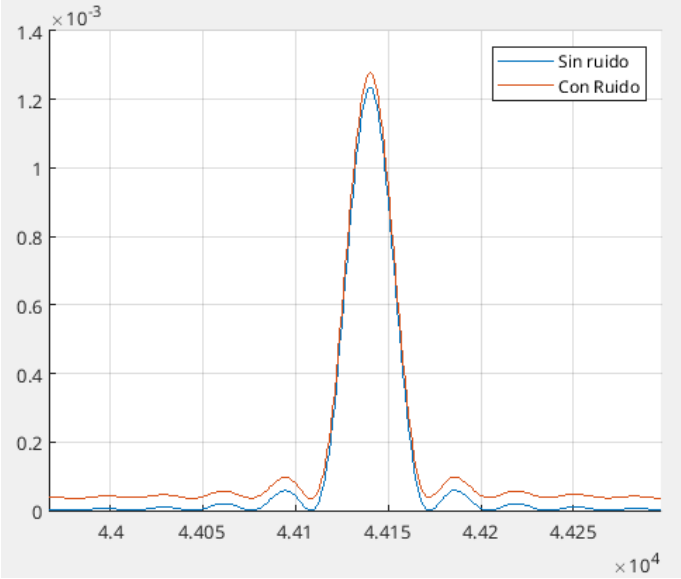
Otro detalle a tener en cuenta es que el piso de ruido tiene valores máximos y mínimos y, dependiendo del valor que se tome, la SNR será distinta. Es entonces que se busca una ventana de valores lejos del impulso (1000 muestras) y se toman una cantidad de muestras para poder promediar el ruido y obtener la potencia.



El código utilizado es el siguiente.

```
[max_val,max_pos] = max(y_mf_accum);              % Valor e índice del pico de la señal
mean_noise = mean(y_mf_accum(max_pos+1000:end)); % Media del ruido
```

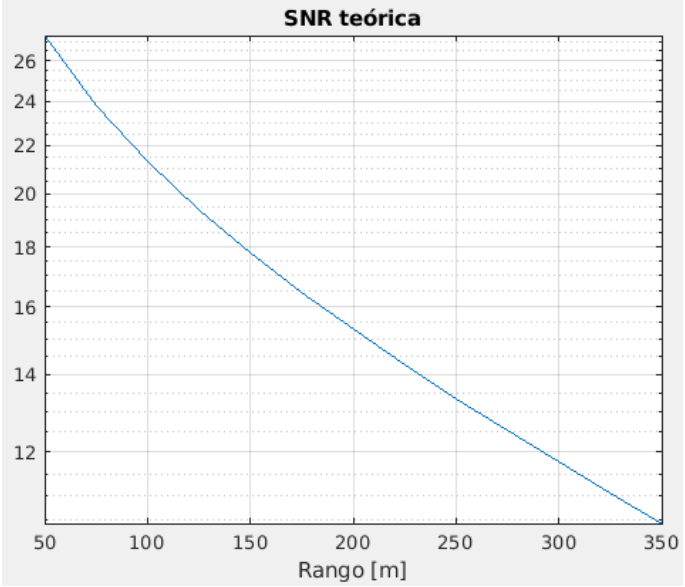
Por último, el piso de ruido hace que el punto máximo del impulso esté más arriba de lo que debería. Por lo tanto, la solución es restarle al punto máximo ese piso de ruido.

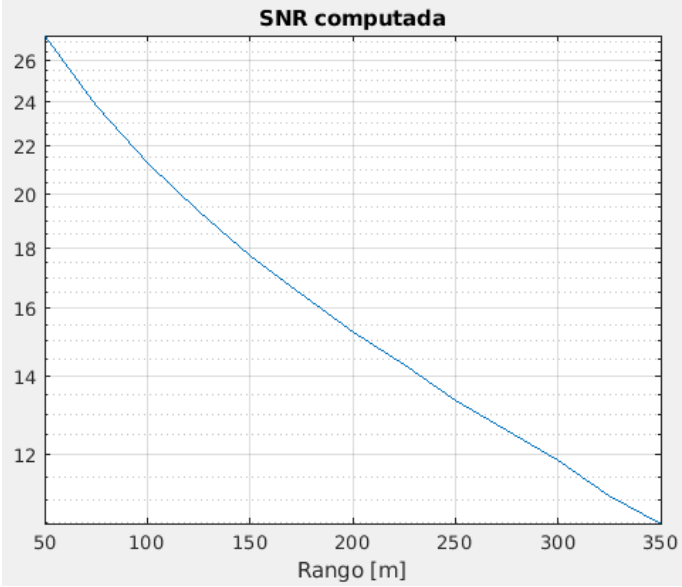


A continuación el código empleado.

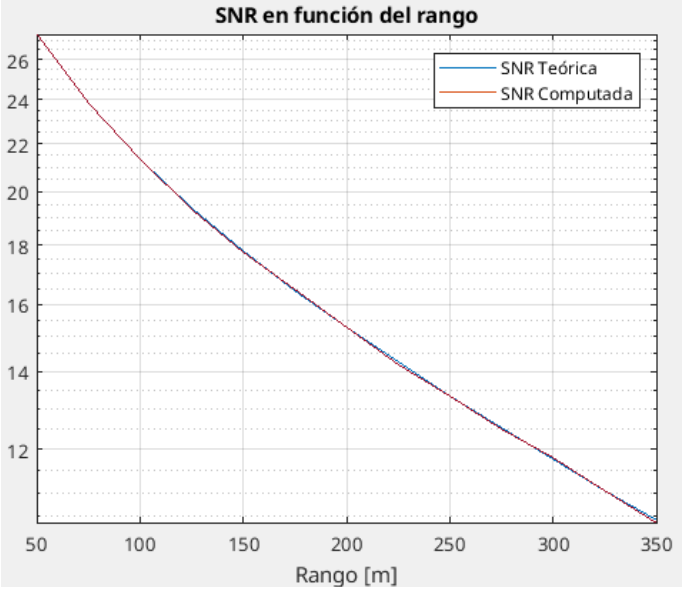
```
snr_comp(j) = (max_val-mean_noise)/mean_noise; % Cálculo de SNR
snr_comp_dB(j) = 10*log10(snr_comp(j));    % SNR logarítmico
```

Las siguientes imágenes muestran los resultados obtenidos. Se puede observar que la SNR teórica y la computada son casi idénticas en la mayoría de los rangos, excepto cuando las distancias se hacen mayores y ya empiezan a diferenciarse. Esto se debe a la atenuación que sufre la potencia recibida a medida que las distancias crecen.

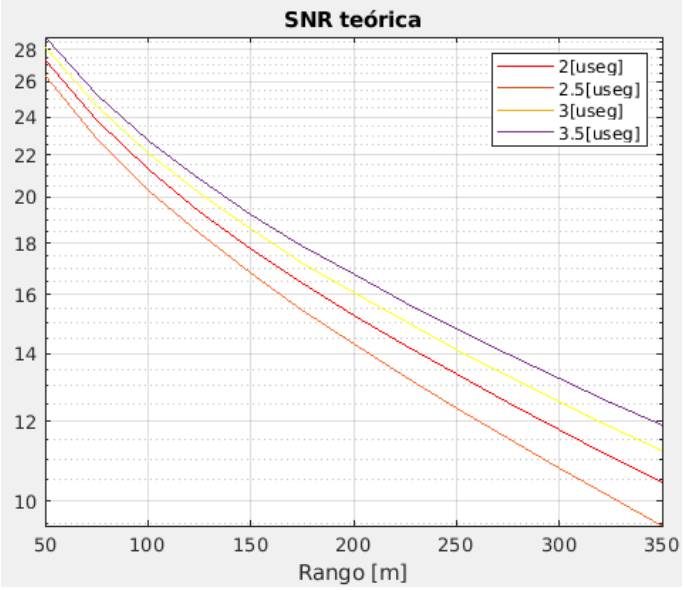




Luego, se superponen ambas gráficas para una mejor comparación.



Por último, se barre la simulación para distintos valores del T_{meas} . Los resultados son los siguientes.



Se puede observar que a medida que aumenta el T_{meas} , también aumenta la SNR, son directamente proporcionales.

Ejercicio 4:

Usando el mismo simulador anterior, computar la Receiver Operation Curve (ROC), usando la explicación vista en clase. Dejar todos los parámetros del LiDAR fijos. Barrer la distancia del target (probar 100m, 150m, 200m, 250m, 300m) y superponer todas las ROCs obtenidas. En base a la SNR teórica de cada caso, computar la ROC teórica con la función rocsnr(SNR,'SignalType',NonFluctuatingNonCoherent') y comparar con la ROC obtenida.

El primer paso fue dividir la salida del matched-filter en celdas de igual tamaño y se toman las muestras que corresponden al centro de cada celda. El código utilizado es el siguiente.

```
fbeat_index = round(fbeat/fs*NFFT)+1;           % Pos. teo. (muestras) del fbeat en la transformada
fft_dec_offset = mod(fbeat_index-1,FFT_NOS);    % Pos. de la celda donde está el máximo
fvec_dec = fvec(1+fft_dec_offset:FFT_NOS:end);  % Vector de frec. decimado, con máximo en medio de
la celda
```

Luego para cada experimento se obtienen los centros de las celdas:

```
y_mf = abs(fft(fe_output, NFFT)).^2;
y_mf_dec = y_mf(1+fft_dec_offset:FFT_NOS:end);
```

El paso siguiente consiste en obtener, de cada experimento, la muestra correspondiente a la celda de interés y guardarla en un vector (`vector_COI`). El índice de la misma en el vector (`y_mf_dec`) está determinado por las siguientes líneas de código.

```
fbeat = chirp_slope*2*real_range/c; % Frecuencia (teórica) del tono resultante
COI = ceil(fbeat*Tmeas);            % Índice de la celda de interés en fvec_dec
vector_COI(i) = y_mf_dec(COI);
```

Las muestras restantes se guardan en un vector (`noise_samples`) que contiene todas las muestras de ruido de todos los experimentos.

```
if vector_inter==0  % Sólo primera iteración
       vector_inter = zeros(Nexp, length(aux));
       vector_inter(i,:) = aux; % Al exp. actual lo guardo en la matriz
   else
       vector_inter(i,:) = aux;
   end
noise_samples = reshape(vector_inter, [], 1);
```

El paso siguiente es decidir cuántos y cuáles serán los valores de los umbrales de decisión. Estos son necesarios para poder obtener múltiples valores de probabilidad de falsa alarma y probabilidad de detección con el fin de poder calcular la ROC. A continuación se tiene el fragmento de código utilizado en esta parte.

```
Nthrs = 1000;   % Cantidad de thresholds
max_threshold = max(vector_COI)*1.05;  % Threshold máximo
min_threshold = max_threshold/Nthrs;    % Threshold mínimo
thresholds = (min_threshold:min_threshold:max_threshold);
```

Para poder calcular la probabilidad de falsa alarma y la probabilidad de detección es necesario tener la información de la cantidad de muestras de ruido que superan o no los distintos umbrales y, lo mismo con las muestras de las celdas de interés. Las cuatro posibilidades existentes son:

- True positive(TP): supera el umbral en la celda de interés.
- False negative(FN): no supera el umbral en la celda de interés.
- False positive(FP): supera el umbral en la celda de muestras de ruido.
- True negative(TN): no supera el umbral en la celda de muestras de ruido.

Se crean vectores para almacenar las ocurrencias de cada uno.

```
fn_vector = zeros(Nthrs,1);   % False Negatives
fp_vector = zeros(Nthrs,1);   % False Positives
tp_vector = zeros(Nthrs,1);   % True Positives
tn_vector = zeros(Nthrs,1);   % True Negatives
```

El código utilizado es el que se muestra a continuación. Se tiene una matriz X donde cada fila corresponde a un umbral y cada columna a una celda COI. La misma contiene un "1" en la coordenada (i,j) si para el experimento j se supera el umbral i, caso contrario se tiene un "0". Esto ocurre de forma análoga en la matriz Y con las muestras de ruido. Por ende un "0" en la matriz X significa un FN y un "1" significa un TP. Lo mismo sucede en la matriz Y donde un "1" corresponde a un FP y un "0" a un TN.

Las comparaciones de las muestras con los thresholds se realizan de la siguiente manera.

```
X = (abs(vector_COI)>thresholds).'; % Ths = filas, exp = columnas
Y = (abs(noise_samples)>thresholds).';
```

El el siguiente bucle se cuentan la cantidad de TP, FN, FP y TN y con esto se calcula la probabilidad de falsa alarma (P_{FA}) y la probabilidad de detección (P_D) para cada umbral.

$$P_{FA} = \frac{FP}{Noise\ Cells * N_{exp}} \qquad P_D = \frac{TP}{N_{exp}}$$

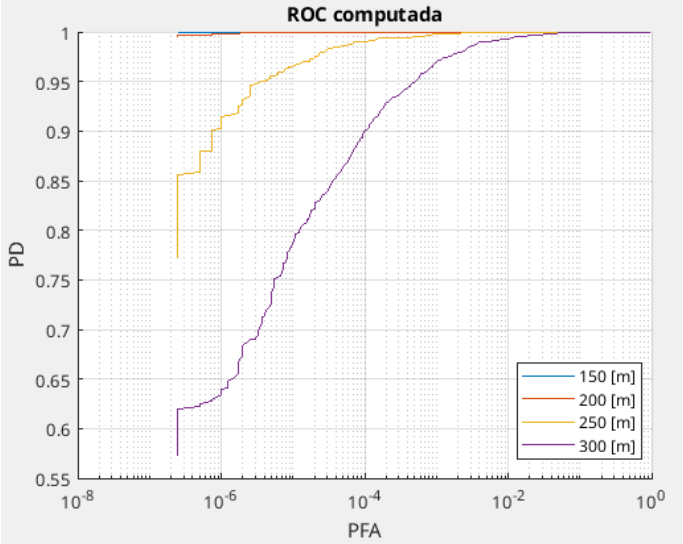
```
ths_stats = zeros(4,length(thresholds)); % TPs, FNs, FPs, TNs = filas en ese orden ; Ths = columnas
for i=1:1:length(thresholds)
   ths_stats(1,i) = sum(X(i,:));                    % TPs para th i
   ths_stats(2,i) = length(X(i,:))-ths_stats(1,i); % FNs para th i
   ths_stats(3,i) = sum(Y(i,:));                    % FPs para th i
   ths_stats(4,i) = length(Y(i,:))-ths_stats(3,i); % TNs para th i

   PD(i) = (ths_stats(1,i))/length(vector_COI);
   PFA(i) = (ths_stats(3,i))/length(noise_samples);
end
```

El gráfico de la ROC se obtiene con la probabilidad de detección (PD) en función de la probabilidad de falsa alarma (PFA). En la consigna se asignan distintos rangos para realizar la ROC y comparar lo que ocurre con ellos.

El cálculo de las ROCs para los distintos rangos se realizó con la siguiente línea de código.

```
semilogx(PFA,PD);grid on;xlabel("PFA");ylabel("PD");title("ROC");
```

Los resultados obtenidos son los siguientes.

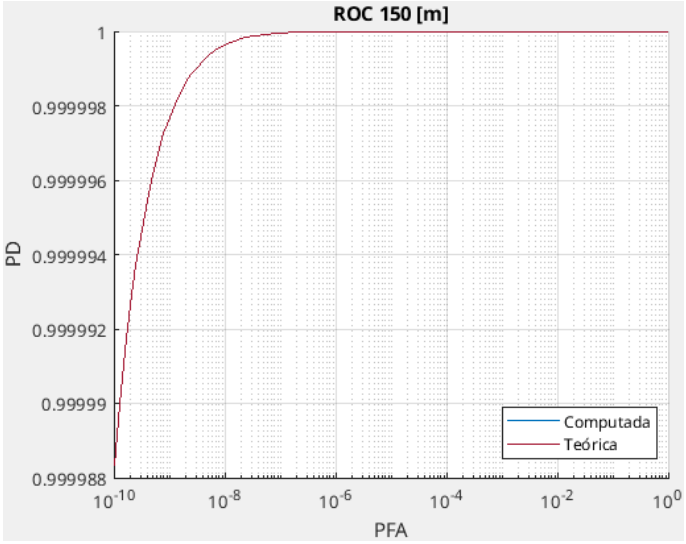


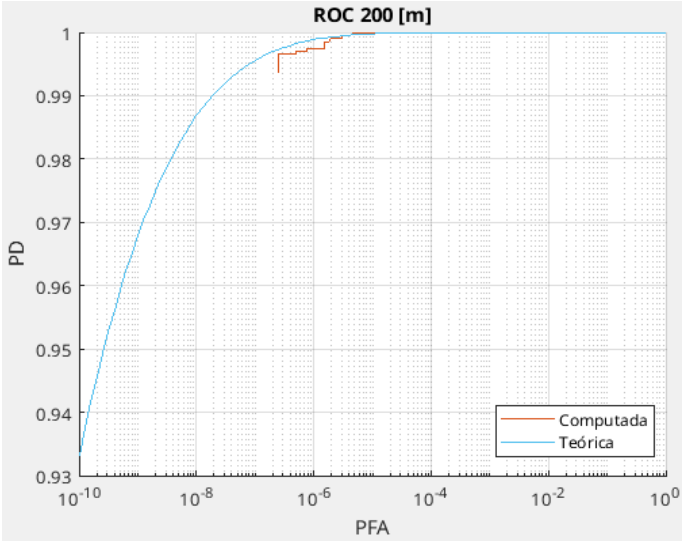
El motivo por el cual el valor mínimo de PFA es 2.5×10^{-7} se debe al tamaño del vector de muestras de ruido, el mismo es de 4 millones de muestras, por lo tanto si sólo una de estas pasa el threshold el valor de PFA será el mencionado anteriormente, y por usar escala logarítmica es imposible representar valores iguales a cero.

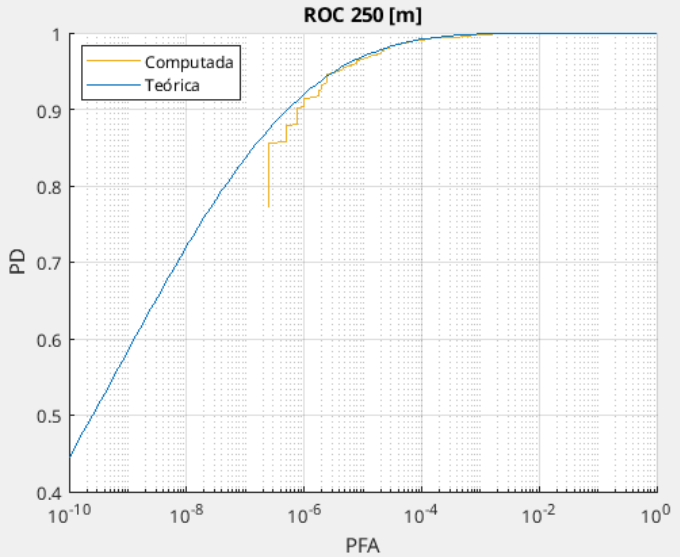
El tamaño del vector de muestras de ruido no podía ser aumentado, esto es debido a que la matriz Y que contiene el resultado de cada threshold en cada muestra alcanzaba el tamaño máximo, limitado por la memoria del dispositivo en el que se corría el script $8GB$.

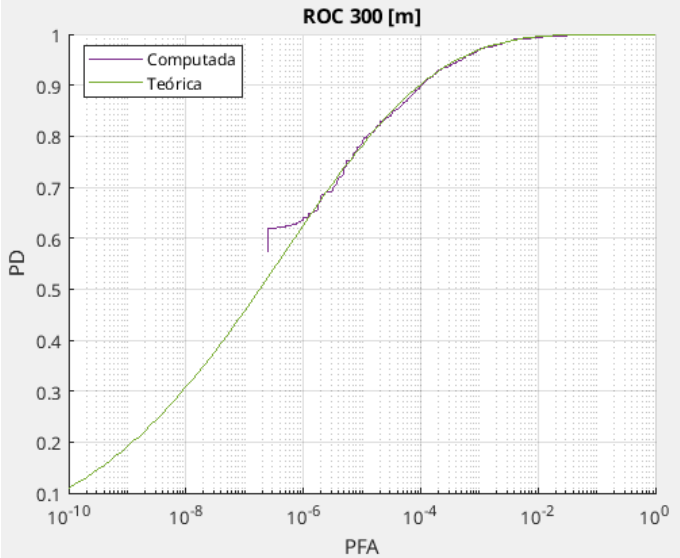
Como último paso se procede a comparar cada una de las ROC computadas con su respectiva ROC teórica obtenida mediante la función *rocsnr()*.

```
[Pd_teo,Pfa_teo] = rocsnr(theo_snr_dB,SignalType='NonFluctuatingNonCoherent');
semilogx(Pfa_teo,Pd_teo);grid on;xlabel("PFA");ylabel("PD");title("ROC");   % ROC computada
```






ROC 250 [m]
Computada
Teórica
PD
PFA
1
0.9
0.8
0.7
0.6
0.5
0.4
10^{-10}
10^{-8}
10^{-6}
10^{-4}
10^{-2}
10^{0}


ROC 300 [m]
Computada
Teórica
PD
PFA
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
10^{-10}
10^{-8}
10^{-6}
10^{-4}
10^{-2}
10^{0}

Bibliografía

- Distribución de Rayleigh
- Probabilidad de Falsa alarma
- Precisión de un Radar
- Principles of Modern Radar: Basic principles
- ROC - Matlab
- Apuntes y códigos vistos en clase.
- Código de todos los ejercicios