UNIVERSIDAD NACIONAL DE CUYO

Instituto Balseiro

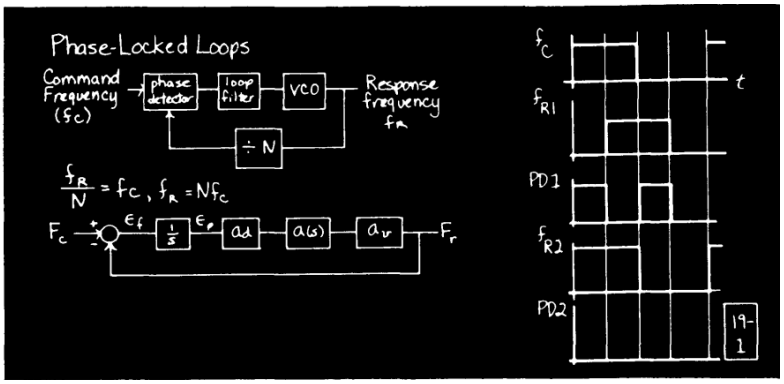
Ingeniería en Telecomunicaciones

Laboratorio II

Lazos enganchados en fase

Manual de prácticas (v1.0)

Ing. L. Horacio Arnaldi[1]



[1] arnaldi@cab.cnea.gov.ar

Índice

1. Introducción

En esta práctica vamos a simular el funcionamiento de un PLL utilizando *Python* como herramienta de cálculo. La estructura de bloques del PLL que vamos a estudiar puede verse en la Figura 1. Como vemos consta de una señal de entrada a la que se aplica un detector con la señal que viene del oscilador

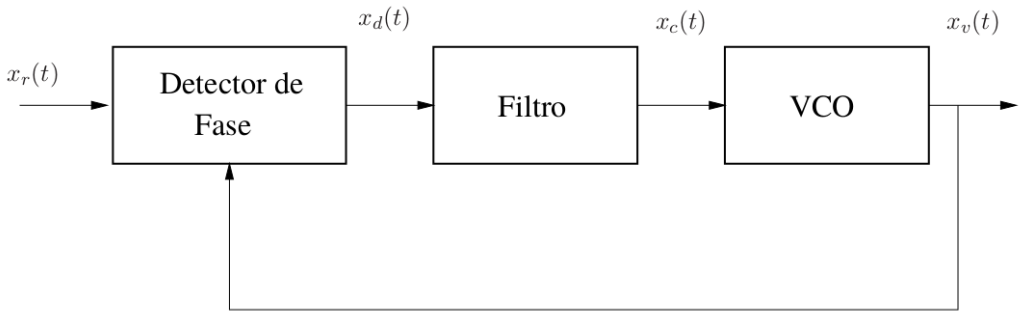


Figura 1: Arquitectura básica de un lazo enganchado en fase.

controlado por tensión (VCO). El resultado se pasa por un filtro paso bajo, y la tensión resultante se aplica al VCO que modificará consecuentemente su frecuencia de oscilación. Para realizar el estudio del PLL, vamos a calcular las señales en los distintos puntos del PLL, en diferentes instantes temporales consecutivos. Es de ir realizaremos una simulación en el dominio del tiempo, para poder comprobar los transitorios de funcionamiento del circuito.

2. Fuente de Señal

Para simular el comportamiento del PLL, deberemos en primer lugar generar la señal de entrada al PLL , $x_r(t)$. Esta señal será una señal sinusoidal tipo seno de una determinada frecuencia, y con una fase inicial establecida.

Ejercicio 1. Construir una función en Python que genere N muestras de la señal de entrada al PLL: **fuente(**f_0**,** T_s**, fase_inicial,** N**)**. Donde f_0 es la frecuencia de la señal, T_s es el periodo entre muestras (periodo de muestreo), fase_inicial es la fase inicial de la señal, y N es el número total de muestras de la señal a generar.

Ejercicio 2. Utilizando la función construida representar la señal de entrada al PLL para $N = 1000$ muestras, $f_0 = 1\,\text{kHz}$, fase_inicial $= \pi/2$, y periodo de muestreo: $T_s = 1/(100 f_0)$.

Con la función construida podemos fácilmente generar una señal de entrada al PLL con un salto de fase en un instante temporal determinado.

Ejercicio 3. Utilizando la función construida representar la señal de entrada al PLL en las mismas condiciones anteriores, pero introduciendo un salto de fase de valor π en la muestra: $N/2$.

3. Detector de fases

El detector de fases más simple consiste en un multiplicador ideal de señales (multiplicador de cuatro cuadrantes).

Ejercicio 4. Construya una función en Python que implemente el detector de fases entre una muestra temporal de la señal de entrada x_r y una muestra temporal de la señal del VCO ($x_v = x_{VCO}$): **detector(**x_r, x_v**)**.

Ejercicio 5. Usando las funciones anteriores construya dos señales de frecuencia ($f_0 = 1\,\text{kHz}$), una con fase inicial cero y la otra con fase inicial ($\pi/4$). Pasar las dos señales por el detector de fases. Tenga en cuenta que debe pasar el detector muestra a muestra con un bucle temporal. Representar la señal resultante en tiempo y frecuencia. Comentar el resultado obtenido. Interpretar el significado de la componente continua obtenida.

Ejercicio 6. Repetir el ejercicio con dos señales, una con fase inicial cero y la otra con fase inicial ($\pi/2$). ¿Qué fase detecta el detector de fase en este caso?. Comentar los resultados obtenidos.

4. Filtro

El siguiente elemento del PLL está formado por un filtro paso bajo, que eliminará el armónico de frecuencia doble que aparece a la salida del detector. La salida del filtro será una tensión continua que será aplicada al VCO. Esta tensión continua hace que el VCO cambie su frecuencia de oscilación.

Vamos ahora a estudiar el funcionamiento de dos de los tres filtros que más se utilizan en PLL comerciales: el filtro *RC* y el *lead-lag activo*. En Python es posible implementar un filtro digital cuando se conoce la función de transferencia en el dominio de Laplace del filtro. Esto se realiza utilizando el comando: *bilinear*, que implementa la transformación bilineal [2]:

$$s = \frac{2}{T_s}\frac{1 - z^{-1}}{1 + z^{-1}} \tag{1}$$

Los argumentos del comando *bilinear* pueden verse en la ayuda de Python. Básicamente hay que especicar dos vectores. El primero es un vector que contiene los coeficientes de las potencias de s del numerador de Laplace. El segundo vector debe contener los coeficientes de las potencias de s del denominador de Laplace. El comando tiene un tercer argumento, que debe ser la frecuencia de muestreo que se utilice para pasar del filtro continuo en el dominio de Laplace al filtro discreto (dominio z). El comando devuelve los coeficientes a_k y b_k del correspondiente filtro digital que implementa la función de transferencia especificada. Estos coeficientes ya pueden utilizarse en el comando *lfilter* [3] para realizar el filtrado de la señal.

La última consideración que hay que tener en cuenta es que se desea realizar un filtrado muestra a muestra de la señal. Como se está realizando un filtrado muestra a muestra, es necesario guardar el estado final del filtro en una variable, después de filtrar cada muestra. Este estado final se especificará como estado inicial del filtro al realizar el filtrado de la siguiente muestra. El uso del comando *lfilter* en este caso puede escribirse de modo genérico así: **lfilter(Num, Den, x, estado_inicial)**.

Ejercicio 7. Utilizando todas las consideraciones anteriores, construir una función en Python que sirva para realizar el filtrado muestra a muestra de la señal de entrada (x_d): **filtro(x_d, estado_inicial, tipo, C, R_1, R_2, T_s)**. La variable (*tipo*) podrá tomar los valores "rc" y "lead-lag activo", para implementar en cada caso la correspondiente función de transferencia.

Para el filtro "rc" y el "lead-lag activo" tomar como parámetros de entrada el coeficiente de amortiguamiento (ξ), y la pulsación propia (w_n). Calcular los restantes componentes del filtro a partir de estos dos parámetros de entrada, suponiendo $R = R_1 = R_2 = C$.

Ejercicio 8. Introducir una función δ como señal de entrada al filtro "rc", con el fin de obtener la respuesta al impulso y la función de transferencia del filtro. Dibujar la respuesta al impulso y la función de transferencia obtenida para los tres casos siguientes: ($w_n = 2$, $\xi = \sqrt{2}/2$), ($w_n = 100$, $\xi = \sqrt{2}/2$) y ($w_n = 2$, $\xi = 5$). Comentar cómo varía en cada caso la respuesta del filtro.

[2] Ver transformación bilineal

[3] Ver función lfilter

Vamos a aplicar estos filtros a la señal de salida del detector de fases. Para ello tomar las dos señales generadas anteriormente con la función **fuente**, una con fase inicial cero y la otra con fase inicial $\pi/4$ (frecuencia portadora: $f_0 = 1\,\text{kHz}$). Pasar las señales por el detector de fases (función **detector**).

Ejercicio 9. Tomando $N = 2000$ muestras, aplicar el filtro "rc" en los tres casos anteriores: ($w_n = 2$, $\xi = \sqrt{2}/2$), ($w_n = 100$, $\xi = \sqrt{2}/2$) y ($w_n = 2$, $\xi = 5$). Representar la señal después del filtro tanto en tiempo como en frecuencia. Para la representación de señales en frecuencia en Python ver el comando *fft* [4]. Comentar las diferencias obtenidas en los tres casos.

Ejericio 10. Repetir el ejercicio 9 pero aplicando el filtro "lead-lag activo", con los valores: ($w_n = 2$, $\xi = \sqrt{2}/2$), ($w_n = 100$, $\xi = \sqrt{2}/2$) y ($w_n = 2$, $\xi = 0{,}1$). Comentar los resultados obtenidos para este filtro.

5. Oscilador Controlado por Voltaje

El último módulo que hay que simular en el PLL es el Oscilador Controlado por Voltaje (VCO). Como sabemos, este oscilador va a generar a la salida una señal sinusoidal del tipo:

$$x_v(t) = \sin(\theta(t)) \tag{2}$$

La frecuencia de esta señal será la frecuencia de una portadora (f_0), con una desviación proporcional a la tensión que exista en la entrada del VCO, es de ir:

$$f(t) = f_0 + K x_c(t) \tag{3}$$

Notar que f_0 es la frecuencia de oscilación natural del VCO, es decir, la frecuencia a la que oscila en ausencia de voltaje de control ($x_c(t) = 0$). Además, K es la ganancia de lazo, y $x_c(t)$ es la tensión a la salida del filtro (ver Figura 1). Por la conocida relación entre la frecuencia y la fase:

$$f(t) = \frac{1}{2\pi}\frac{d\theta(t)}{dt} \tag{4}$$

podemos escribir la fase instantánea del VCO de la siguiente forma (utilizamos las ecuaciones (3) y (4)):

$$\theta(t) = 2\pi \int f(t)dt = 2\pi f_0 t + 2\pi K \int x_c(t)dt \tag{5}$$

Para realizar el análisis del PLL muestra a muestra, necesitamos conocer la fase de la señal del PLL en el instante temporal siguiente, a partir de la fase del PLL en el instante temporal actual. Si T_s es el periodo de muestreo tendremos:

$$\theta(t + T_s) = 2\pi f_0(t + T_s) + 2\pi K \int_0^{(t+T_s)} x_c(\tau)d\tau \tag{6}$$

$$= 2\pi f_0 t + 2\pi K \int_0^{t} x_c(\tau)d\tau + 2\pi f_0 T_s + 2\pi K \int_0^{(t+T_s)} x_c(\tau)d\tau \tag{7}$$

utilizando la aproximación del rectángulo para la segunda integral:

$$\int_t^{(t+T_s)} x_c(\tau)d\tau = x_c(t)T_s \tag{8}$$

[4] Ver función FFT

La fase instantánea en el instante $(t + T_s)$ queda entonces:

$$\theta(t + T_s) = 2\pi f_0 t + 2\pi K \int_0^t x_c(\tau)d\tau + 2\pi[f_0 + Kx_c(t)]T_s \tag{9}$$

Utilizando las ecuaciones (3) y (5), obtenemos finalmente:

$$\theta(t + T_s) = \theta(t) + 2\pi f(t)T_s \tag{10}$$

Ejercicio 11. En base al desarrollo anterior, construir una función que, dada la frecuencia actual $(f = f_0 + Kx_c(t))$ y la fase en el instante t, calcule la fase en el instante siguiente $(t+T_s)$: **fase_final = fase(T_s,f,fase_inicial)**. Al construir la función, asegurarse que la fase de salida estará comprendida entre el intervalo $(0, 2\pi)$.

Con la función construida, es posible simular el comportamiento del VCO en cada muestra de la siguiente forma:

```
ph = fase(T_s,f0 + K x_c,ph)   #Calcula la fase en el siguiente instante.
V_vco(i) = sin(ph)             #Señal de salida del VCO.
```

siendo (x_c) la tensión de control al VCO (ver Figura 1). El código mostrado estará dentro del bucle que barre todas las muestras de la señal (índice i).

La aplicación mas inmediata de la función que acabamos de construir es la implementación de un modulador de FM. Vamos a tomar una frecuencia central de oscilación del VCO de: $f_0 = 1\,\mathbf{kHz}$. Tomaremos como señal moduladora una señal sinusoidal de frecuencia: $f_m = 40\,\text{Hz}$. Tomar para este análisis una frecuencia de muestreo de: $f_s = 10f_0$, con un número total de muestras: $N = 1000$.

Ejercicio 12. Generar en primer lugar la señal moduladora (o señal de control, $x_c(t)$) usando la función "**fuente**" ya construida (tomar como fase inicial cero). Representar la señal en tiempo y frecuencia.

Ejercicio 13. Obtener la señal a la salida del VCO sin introducir señal de control ($K = 0$). Representar la señal en tiempo y frecuencia. Comentar el resultado obtenido.

Ejercicio 14. Obtener la señal a la salida del VCO introduciendo la señal de control con una ganancia $K = 100$. Dibujar la señal obtenida en tiempo y frecuencia. ¿Ha obtenido la señal modulada en frecuencia?.

6. Lazo Enganchado en Fase

Utilizando todas las funciones realizadas hasta ahora, construir un programa que simule el funcionamiento de un PLL. Vamos a simular el funcionamiento de un PLL, cuya frecuencia de oscilación libre es de $f_0 = 1\,\text{kHz}$. La señal de entrada al PLL será una señal de la misma frecuencia (sin error de frecuencia), y con error de fase inicial de $\pi/2$. Tomar además $N = 3000$ muestras y una frecuencia de muestreo de $f_s = 100f_0$. En estas condiciones realizar los siguientes experimentos.

Ejercicio 15. Utilizando el filtro "rc", ajustar los parámetros del bucle a ($\xi = \sqrt{2}/2$, $w_n = 180$). Representar la señal a la salida del filtro con el fin de comprobar el tipo de amortiguamiento del bucle. Repetir con un factor de amortiguamiento de ($\xi = 0{,}3$) y ($\xi = 2{,}0$) para comprobar cómo cambia el amortiguamiento del bucle con el parámetro de amortiguamiento ξ. Representar todas las gráficas.

Ejercicio 16. Para los tres casos anteriores representar el diagrama de Lissajous y la gráfica de error entre la señal de salida del VCO y la señal de entrada. ¿Consigue engancharse el PLL en fase?.

Ejercicio 17. Para comprobar como afecta el ancho de banda del filtro, repetir el ejercicio con $\xi = \sqrt{2}/2$, $w_n = 120$). ¿Qué ocurre al disminuir el ancho de banda del filtro?.

Ejercicio 18. Utilizando el filtro "lead-lag activo", comprobar el enganche del PLL para unos parámetros de bucle: ($\xi = \sqrt{2}/2$, $w_n = 40$).

Ejercicio 19. Comprobar el enganche del filtro si se aumenta el ancho de banda a: ($\xi = \sqrt{2}/2$, $w_n = 120$). ¿Se comporta el PLL igual que antes con el ancho de banda del filtro?.

6.1. Errores de fase

A la señal anterior, introducir en la muestra ($N/2$) un error de fase de ($\pi/4$). Dibujar en cada caso la señal de error y el diagrama de Lissajous.

Ejercicio 20. Con el filtro "rc", ajustar los parámetros del bucle a: ($\xi = 2{,}0$, $w_n = 220$). ¿Consigue engancharse el PLL al introducir el error de fase?.

Ejercicio 21. Repetir el proceso para unos parámetros: ($\xi = \sqrt{2}/2$, $w_n = 220$). Comentar las diferencias que observa.

Ejercicio 22. Con el filtro "lead-lag activo", ajustar los parámetros del bucle a: ($\xi = \sqrt{2}/2$, $w_n = 80$). ¿Se engancha correctamente el PLL?.

Ejercicio 23. Repetir el proceso para unos parámetros: ($\xi = \sqrt{2}/2$, $w_n = 180$). ¿Qué ocurre con la velocidad de enganche del PLL?.

Vamos a trabajar ahora con una señal similar a la anterior, pero que presenta en la muestra $N/2$ un error de fase de π. Igual que antes representar la señal de error y el diagrama de Lissajous que se obtiene en cada caso.

Ejercicio 24. Con el filtro "rc", ajustar los parámetros del bucle a: ($\xi = 2{,}0$, $w_n = 220$). ¿Consigue engancharse el PLL al introducir el error de fase?. ¿Porqué?.

Ejercicio 25. Con el filtro "lead-lag activo", ajustar los parámetros del bucle a: ($\xi = \sqrt{2}/2$, $w_n = 180$). ¿Se engancha correctamente el PLL?. ¿Porqué cree que ocurre esto?.

6.2. Errores de frecuencia

Generar ahora una señal de entrada al PLL con el error de fase inicial de $\pi/2$, y un error de frecuencia del 10 %. Representar en cada caso la señal de error y el diagrama de Lissajous.

Ejercicio 26. Con el filtro "rc", ajustar los parámetros del bucle a: ($\xi = \sqrt{2}/2$, $w_n = 220$). ¿Consigue engancharse el PLL al introducir el error de frecuencia?.

Ejercicio 27. Repetir el experimento tomando los siguientes parámetros de bucle: ($\xi = \sqrt{2}/2$, $w_n = 450$). Interprete el resultado obtenido.

Ejercicio 28. Repetir el experimento tomando los siguientes parámetros de bucle: ($\xi = \sqrt{2}/2$, $w_n = 450$). Interprete el resultado obtenido. En concreto, ¿qué sucede con el error de fase del PLL?, ¿porqué?.

Ejercicio 29. Con el filtro "lead-lag activo", ajustar los parámetros del bucle a: ($\xi = \sqrt{2}/2$, $w_n = 180$). ¿Se engancha correctamente el PLL?.

Ejercicio 30. Repetir el ejercicio, si reduce la pulsación propia a $w_n = 120$, ¿se engancha el PLL correctamente?. ¿Por qué?.

Ejercicio 31. Repetir el ejercicio, si reduce la pulsación propia a $w_n = 80$, ¿se engancha el PLL correctamente?. ¿Por qué?

Ejercicio 32. Para comprobar si el problema es el margen de enganche, reduzca el error de frecuencia al 3 %. ¿Se engancha ahora correctamente el PLL?. ¿Cual es la relación entre el margen de enganche y el ancho de banda del filtro?.

6.3. Rampa en frecuencia

Vamos a comprobar el comportamiento del PLL ante una rampa en frecuencia. Como siempre representar la señal de error y el diagrama de Lissajous.

Ejercicio 33. Construir una función en Python que genere la señal de entrada al PLL, constituida por una rampa en frecuencia: **fuente_fvar(**f_0**,** $f_{\mathbf{final}}$**,** T_s**,** $\mathbf{fase_{inicial}}$**,** N**)**.

Con la función que acaba de crear, genere una señal con error de fase inicial ($\pi/2$), y una rampa en frecuencia del 10 %, tomando como frecuencia inicial: $f_0 = 1\,\text{kHz}$. Vamos a tomar esta señal como entrada al PLL.

Ejercicio 34. Con el filtro "rc", ajustar los parámetros del bucle a: ($\xi = \sqrt{2}/2$, $w_n = 450$). ¿Consigue engancharse el PLL al introducir el error de frecuencia en forma de rampa?.

Ejercicio 35. Con el filtro "lead-lag activo", ajustar los parámetros del bucle a: ($\xi = \sqrt{2}/2$, $w_n = 180$). ¿Se engancha correctamente el PLL?. ¿Existe un error de fase constante?.

Ejercicio 36. Repetir el ejercicio tomando: ($\xi = \sqrt{2}/2$, $w_n = 280$). ¿Qué sucede con el error de fase del PLL?, ¿por qué?.

6.4. Demodulación Coherente de una Señal

Vamos a simular ahora el comportamiento de los PLL's diseñados, cuando se utilizan para realizar la demodulación coherente, en un receptor, de una señal digital transmitida con modulación en amplitud ASK. El esquema del sistema de comunicaciones completo puede verse en la Figura 2. Como se aprecia,

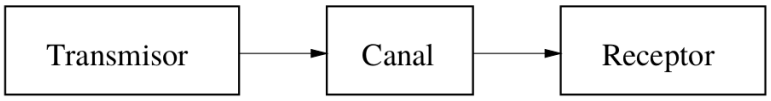


Figura 2: Esquema del sistema de comunicaciones.

el sistema de comunicaciones consta de un transmisor, el canal por donde viaja la información, y el receptor que recibe y recupera el mensaje enviado.

La estructura del transmisor puede verse en la Figura 3. Como se aprecia, consta en primer lugar de

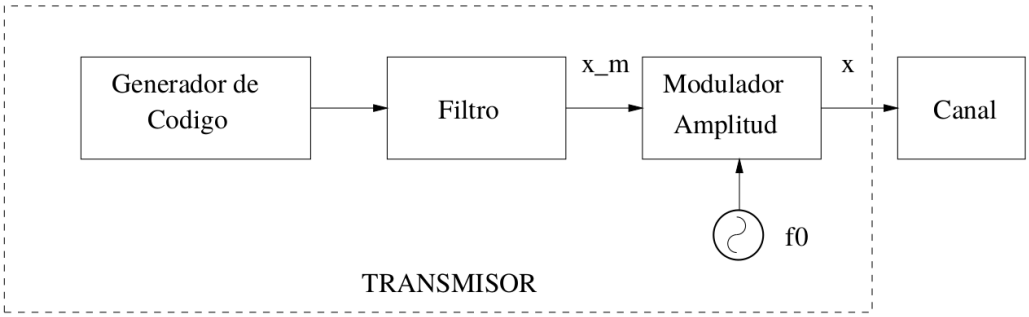


Figura 3: Estructura del transmisor.

un generador de código encargado de generar la información o pulsos digitales. Seguidamente metemos un filtro encargado de optimizar la forma del pulso al canal que va a utilizarse como medio de transmisión. Finalmente se mete un modulador de portadora encargado de generar la señal de radiofrecuencia (f_0) modulada en amplitud para ser transmitida por el canal.

Para simular el funcionamiento del transmisor vamos a crear una función en Python con la siguiente estructura: **emisor(**N_d**,** f_0**,** d_t**)**. Esta función entregará dos salidas: x y x_m, donde x es la señal de

salida modulada, x_m es la señal moduladora, N_d es el número total de pulsos transmitidos, f_0 es la frecuencia de la portadora, y hemos llamado d_t al periodo de muestreo.

Para generar la señal en el transmisor definir una frecuencia de muestreo interna en el transmisor de $f_s = 200$ Hz. Para generar la señal a la salida del Generador de Código, podemos utilizar el siguiente comando: **round(rand)** [5], que da un número entero que puede tomar el valor de cero o uno de forma aleatoria. Con el fin de evitar interferencia entre símbolos, dejaremos un intervalo entre símbolos: $T_b = 2$ seg.

Ejercicio 37. Con las consideraciones anteriores, generar y representar la señal a la salida del Generador de Código de la Figura 3, cuando generamos un número de pulsos igual a: $N_d = 10$.

Ya hemos dicho que la función del filtro es adaptar la señal pulsada al canal de transmisión, tratando de minimizar la interferencia entre símbolos. Sabemos que una forma de minimizar la interferencia entre símbolos es utilizando un filtro en coseno alzado [6]. Existe un comando en Python que permite calcular la respuesta al impulso de un filtro en coseno alzado: $h = \text{rcosfilter}(N, \alpha, T_s, F_s)$, siendo T_s el período de símbolos en segundos, y F_s la frecuencia de muestreo de la señal (tomar $F_s = 1$ Hz).

Ejercicio 38. Con el procedimiento descrito representar la respuesta al impulso y la función de transferencia del filtro en coseno alzado diseñado.

Ejercicio 39. Obtener y dibujar la señal moduladora (x_m) a la salida del filtro.

Ejercicio 40. Repetir el ejercicio anterior tomando un intervalo entre símbolos: $T_b = 1$ seg. ¿Qué ocurre cuando hay dos pulsos consecutivos?. ¿Cuál sería la única forma de evitar interferencia entre símbolos en este caso?.

Volver al ($T_b = 2$ seg) para evitar interferencia entre símbolos.

Ejercicio 41. Obtener y dibujar la señal AM a la salida del transmisor, x (tomar índice de modulación, $m = 1$, y frecuencia de portadora: $f_0 = 1\,\text{kHz}$).

Ahora supondremos que la frecuencia nominal a la que trabaja tanto el transmisor como el receptor es de: $f_0 = 1\,\text{kHz}$. Sin embargo, el oscilador del transmisor presenta una deriva en frecuencia, por lo que la señal emitida por el transmisor tendrá una frecuencia de portadora de: $1{,}05 f_0$.

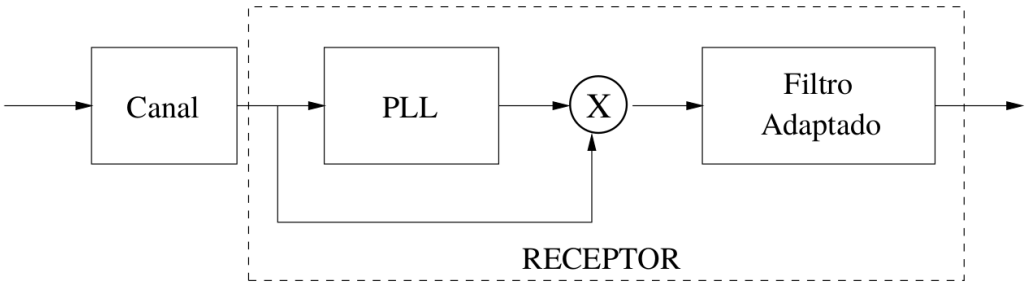


Figura 4: Estructura del receptor cpherente.

El esquema del receptor coherente puede verse en la Figura 4. Como se aprecia, el receptor utiliza un PLL para recuperar la frecuencia y fase de la portadora. A continuación se produce la detección coherente entre la señal recibida y la portadora recuperada por el PLL. Finalmente se produce el filtrado, que en este caso se realiza mediante un filtro adaptado para optimizar la relación señal-ruido. En nuestro caso el filtro adaptado será también un coseno alzado, y su respuesta al impulso puede ser calculada de la siguiente forma: $h = 1 \times 10^{-2} * rcosfilter(1, 200)$.

Realizar la simulación del receptor incorporando los PLL's diseñados. En cada uno de los casos

[5] Ver round y rand

[6] Ver por ejemplo Raised Cosine in Python

siguientes dibujar la señal portadora recuperada por el PLL, el diagrama de Lissajous, la señal a la salida del detector, y la señal a la salida del filtro adaptado.

Ejercicio 42. Intentar realizar la detección coherente con el filtro "rc", tomando los siguientes parámetros: ($\xi = \sqrt{2}/2$, $w_n = 120$). ¿Consigue engancharse el PLL?, ¿por qué?. ¿Recupera correctamente el mensaje?.

Ejercicio 43. Repetir el intento aumentando el ancho de banda del filtro "rc": ($\xi = \sqrt{2}/2$, $w_n = 220$). ¿Consigue engancharse ahora el PLL?. ¿Recupera correctamente el mensaje?.

Ejercicio 44. Repetir el intento con un ancho de banda todavía mayor: ($\xi = \sqrt{2}/2$, $w_n = 450$). ¿Consigue engancharse ahora el PLL?. ¿Recupera correctamente el mensaje?, ¿por qué?.

Ejercicio 45. Intentar realizar la detección coherente con el filtro "lead-lag activo", tomando los siguientes parámetros: ($\xi = \sqrt{2}/2$, $w_n = 120$). ¿Consigue engancharse el PLL?, ¿por qué?. ¿Recupera correctamente el mensaje sin ningún tipo de distorsión?.

Referencias

[1] Floyd M. Gardner, *Phaselock Techniques*, Third Edition, John Wiley and Sons, Inc., Hoboken, New Jersey 2005