Comunicaciones Inalámbricas

Laboratorio 5

Alineación en Fase

1. Introducción

En este laboratorio empezaremos a preparar nuestro receptor para trabajar en un entorno un poco más realista. En particular, atacaremos el problema de la alineación en fase, y experimentaremos con los métodos disponibles con este fin dentro de GNU Radio. Con este fin, partiremos de lo realizado en el laboratorio anterior, e implementaremos dos técnicas distintas de alineación en fase. De aquí en más utilizaremos el pulso SRRC por sus características espectrales.

2. Transmisor

Tal como fue discutido en el teórico, todos los métodos de alineación en fase tienen algún grado de ambigüedad de fase, por lo que es necesario algún método para subsanar este problema. En esta implementación utilizaremos la técnica de codificación diferencial, fundamentalmente debido a su sencillez y eficacia. Con este fin utilizaremos el bloque `Differential Encoder`.

Lea la documentación de este bloque y sobre todo observe que tanto la salida como la entrada son del tipo `Byte`. Asuma un valor de `Modulus` de 4.

1. Escriba una tabla con la salida para cada posible pareja de entrada y salida anterior. La figura 1 muestra un diagrama del codificador (siendo δ_k y s_k la salida del bloque y el símbolo (par de bits en este caso) de entrada respectivamente).

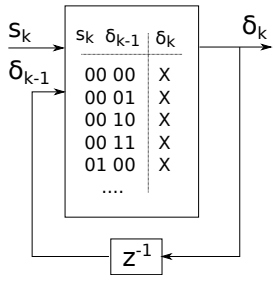
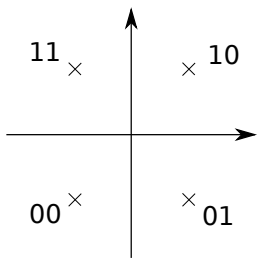


Figura 1: Codificador Diferencial incluyendo algunos valores de entrada.



(a) Constelación a utilizar

$s(k)$	$\Delta\alpha$
00	0
01	$\pi/2$
10	π
11	$-\pi/2$

(b) Tabla de codificación en fase.

Figura 2: Especificación de la codificación diferencial.

2. Utilizando la tabla realizada en la parte anterior verifique que con un `Chunks to Symbols` a la salida del bloque `Differential Encoder` con una constelación como la de la figura 2a se consigue codificar la pareja de bits en la diferencia de fase entre símbolos complejos sucesivos como lo muestra la tabla 2b.

3. Haga los cambios necesarios en el transmisor realizado en el laboratorio anterior para implementar el modulador DQPSK resultante.

3. Canal

El canal ahora no será un mero "pasa-todo", sino que distorsionará la fase de la señal. Por el momento, generaremos un cambio de fase constante.

4. Agregue un bloque `QT GUI Range` que varíe entre $-\pi$ y π, cuyo valor por defecto sea 0 y cuyo `ID` sea `fase`. Este bloque agrega un "slider" a la interfaz gráfica que asigna (en tiempo de ejecución) un valor a la variable indicada en `ID`. Es decir, muy similar al bloque `Variable` pero con la posibilidad de cambiar el valor en tiempo de ejecución.

5. Modifique el parámetro `Taps` del bloque `Channel Model` de tal manera que genere un cambio de fase igual a `fase`. Recuerde que puede utilizar el módulo `math` de `Python` mediante un bloque `Import` con la línea `import math`.

4. Receptor

Lo primero que haremos en recepción es verificar el efecto del cambio en fase sobre la señal recibida.

6. Agregue un bloque `QT GUI Constellation Sink` a la salida del sub-sistema de muestreado implementado en el laboratorio anterior. Verifique que ahora obtiene los símbolos complejos girados un ángulo igual a `fase`.

Exploraremos ahora dos posibles maneras de implementar un decodificador diferencial: mediante un bloque `Costas Loop` y un bloque `Differential Phasor`.

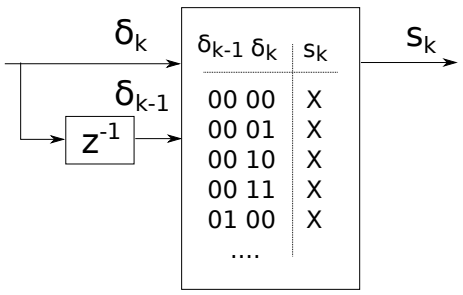


Figura 3: Decodificador diferencial incluyendo algunos valores de entrada.

4.1. Costas Loop

7. Coloque un bloque `Costas Loop` entre el muestreador y el `Constellation Decoder`. Observe qué efecto tiene esto sobre los símbolos complejos mediante otro bloque `QT GUI Constellation Sink`. Verifique que el bloque asume implícitamente una constelación con la misma forma que la que consideramos en la figura 2a. Lea atentamente la documentación del bloque para obtener ayuda sobre cómo asignar los dos parámetros del `Costas Loop`.

8. Para pasar de complejos a símbolos en este caso tendremos que recurrir a la construcción un poco más general del bloque `Constellation Decoder`. El argumento que requiere el bloque será `digital.constellation_calcdist(<constelacion>, [], 4, 1).base()`. Busque la documentación del método `digital.constellation_calcdist` para más ayuda sobre los parámetros.

Lo que estaría restando implementar es el Decodificador Diferencial. Para ello utilizaremos el bloque `Differential Decoder`.

9. Lea la documentación del bloque. Verifique que con un valor de 4 para el parámetro `Modulus` revierte la codificación diferencial realizada en el transmisor. Quizá le ayude realizar una tabla como la de la figura 3.

4.2. Fasor Diferencial

Ahora exploraremos un método que en vez de realizar la codificación diferencial a nivel de bits (o pares de bits en este caso), lo aplica a nivel de los símbolos complejos. Seguramente le convenga guardar una segunda versión de su flow-graph para explorar este método alternativo al Costas Loop.

10. Agregue un bloque `Differential Phasor`. Lea atentamente la documentación del mismo y conéctelo entre el canal y el `Constellation Decoder`. ¿Qué obtiene a la salida de este bloque? Verifique entonces que con la constelación que estábamos utilizando en recepción hasta ahora y el `Differential Decoder` tenemos una gran tasa de error.

11. ¿Qué constelación tenemos que utilizar ahora en recepción? ¿Hace falta el bloque `Differential Decoder`?

4.3. ¿Se necesita alineamiento en fase?

En esta parte verificaremos para qué se necesita alinear en fase, y no basta con la codificación diferencial a nivel de bits. Para ello, tomaremos la versión del flow-graph que utiliza un Costas Loop y el bloque `Differential Decoder`.

12. Quite el bloque `Costas Loop` (recuerde que puede deshabilitar un bloque seleccionándolo y presionando la letra 'd' o con botón derecho y seleccionando 'Disable'). Pruebe distintos valores de `fase` y verifique que el sistema sigue funcionando correctamente.

13. Ahora pruebe con un error de fase igual a $\pi/4$. ¿Porqué el sistema dejó de funcionar? Verifique que si el ruido es moderado este efecto se da incluso cuando el error en fase no es exactamente igual a $\pi/4$.

14. Discuta porqué el método de alineamiento en fase que utiliza el bloque `Differential Phasor` o agregar el `Costas Loop` hace que el sistema sea inmune a este problema.

4.4. Efectos de un corrimiento en frecuencia

Tome valores pequeños (no más de 0.02) para el parámetro `Frequency Offset` del bloque `Channel Model`. Puede agregar un `QT GUI Range` para corroborar su efecto en tiempo de ejecución.

15. ¿Qué efecto tiene esto sobre el espectro de la señal que entra al receptor?

16. ¿Qué efecto tiene sobre los símbolos complejos que salen del filtro apareado?

17. Encuentre experimentalmente qué corrimiento en frecuencia soportan los métodos de alineamiento en fase implementados. En el caso del `Costas Loop` encuentre un valor razonable para el `Loop Bandwidth`.

18. Argumente porqué, si el corrimiento en frecuencia es demasiado grande, es necesario corregirlo **antes** del filtro apareado.

5. Algunos comentarios finales

- Dependiendo del tamaño de los filtros puede ser que el sistema completo comience a ser demasiado para su PC. Le recomendamos que no se exceda en el valor de muestras por símbolos (`samp_per_sym` en la nomenclatura del laboratorio pasado) ni en el largo del filtro SRRC (`len_sym_srrc`). Con un valor de 3 para el primero y de más de 5 para el segundo será suficiente a priori.

- También es importante notar que el audio puede no escucharse bien por varios motivos, siendo algunos ejemplos: saturó la tarjeta de audio (si los valores que entran al bloque `Audio Sink` son muy grandes), la tasa de muestras es menor o mayor a lo indicado, o el proceso es muy demandante a nivel de CPU (como se discutió en el punto anterior).