Teoría de las Comunicaciones 2017 - Trabajo Práctico de Simulación

Adrián Pardini - 54024

Parte 2 - Codificación de Canal

El objetivo del presente trabajo consiste en comparar el desempeño de un sistema de comunicación en canal AWGN con y sin codificación.

Ejercicio 1

Para un código de bloque (9, 5) la matriz generadora en su forma sistémica es:

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & X & X & X & X \\ 0 & 1 & 0 & 0 & 0 & X & X & X & X \\ 0 & 0 & 1 & 0 & 0 & X & X & X & X \\ 0 & 0 & 0 & 1 & 0 & X & X & X & X \\ 0 & 0 & 0 & 0 & 1 & X & X & X & X \end{pmatrix}$$

Donde X son bits que definen al código.

Descartando la fila nula e ignorando todas las permutaciones posibles de un mismo grupo de 5 filas entre las 15 posibles, la cantidad de códigos (9, 5) sistémicos son:

$$n = \frac{15!}{5!(15-5)!} = 3003$$

Si se incoporan todas las permutaciones la cantidad es:

$$n = \frac{15!}{(15-5)!} = 360360$$

En ambos casos el número es lo suficientemente bajo como para considerar un enfoque de fuerza bruta al buscar un código con la mayor distancia mínima posible.

Con ese fin se escribió un script en Python:

```
#!/usr/bin/env python3

"""Finds all the generator matrixes of a (9,5) code and their minimum weight
"""

import itertools
from collections import defaultdict
import numpy as np

I = np.identity(5, dtype=int)
```

```
# All the valid rows of the generator matrix excluding the all zeros
valid_rows = [ [int(x) for x in np.binary_repr(row, 4)] for row in range(1, 16)]
# All permutations of the generator matrix excluding the Identity half
all_codes = itertools.permutations(valid_rows, 5)


# The input alphabet used to measure the code weight, excluding the zero.
input_alphabet = [ [int(x) for x in np.binary_repr(row, 5)] for row in range(1, 32)]
# maps weight -> list of codes
code_weights = defaultdict(list)


def measure_weight (code):
   """Given a code as an 4x5 array returns the minimum weight of all the codewords
    """

    # Build a systemic generator matrix
    G = np.hstack([I, code])
    codewords = np.mod(np.dot(input_alphabet, G), 2)
    weights = np.sum(codewords, axis=1)

    return int(min(weights))


for code in all_codes:
    code = np.array(code)
    weight = measure_weight(code)
    code_weights[weight].append(code)


if __name__ == '__main__':
    for weight, codes in code_weights.items():
     print ('{0:6} codes of minimum distance {1}'.format(len(codes), weight))

    max_weight = max(code_weights.keys())
    code = code_weights[max_weight][0]
    G = np.hstack([I, code_weights[max_weight][0]])
    H = np.hstack([code.transpose(), np.identity(4, dtype=int)])
   print('One code of minimum distance {} is:'.format(max_weight))
    print('G: ')
    print(G)
    print('H: ')
    print(H)
```

Que dió como resultado:

```
304920 codes of minimum distance 2
 55440 codes of minimum distance 3
One code of minimum distance 3 is:
G:
[[1 0 0 0 0 0 0 1 1]
 [0 1 0 0 0 0 1 0 1]
 [0 0 1 0 0 0 1 1 0]
 [0 0 0 1 0 0 1 1 1]
 [0 0 0 0 1 1 0 0 1]]
H:
[[0 0 0 0 1 1 0 0 0]
 [0 1 1 1 0 0 1 0 0]
 [1 0 1 1 0 0 0 1 0]
 [1 1 0 1 1 0 0 0 1]]
```

Empleando este código se podrán detectar hasta 2 errores y corregir un máximo de

$$t = \left\lfloor \frac{d_{min} - 1}{2} \right\rfloor = 1 \; error$$

La ganancia asintótica del código con decisión dura es (Proakis, Digital Communications 8.1)

$$G_a = 10log(R_c \frac{d_{min} + 1}{2}) \; [dB]$$
$$G_a = 1{,}05dB$$

Para un código de bloque (n,k) en un canal simétrico con probabilidad de error p, la probabilidad de error de palabra (corregida) está acotada por la probabilidad de que t o más bits estén errados en un bloque de n bits.

$$Pe_w \leq \sum_{m=t+1}^{n} \binom{n}{m} p^m (1-p)^{(n-m)}$$

Si el código se utiliza sólo como detector la probabilidad de error de palabra está dada por la probabilidad de que $d_{min} - 1$ o más bits estén errados en un bloque de k bits.

$$Pd \leq \sum_{m=d_{min}-1}^{n} \binom{n}{m} p^m (1-p)^{(n-m)}$$

De no emplearse un código corrector la probabilidad de error en un bloque de k bits es:

$$Pe_u = 1 - (1 - p)^k$$

Se procedió en forma similar a la primer parte a relevar las características del sistema, modificando las condiciones de corte para detener cada iteración cuando por cada relación señal a ruido de 0 dB a 10 dB se llegue a 40 errores de palabra luego de emplear el código como corrector o la probabilidad de error de palabra corregida sea menor que $2 \cdot 10^{-5}$, lo que suceda primero.

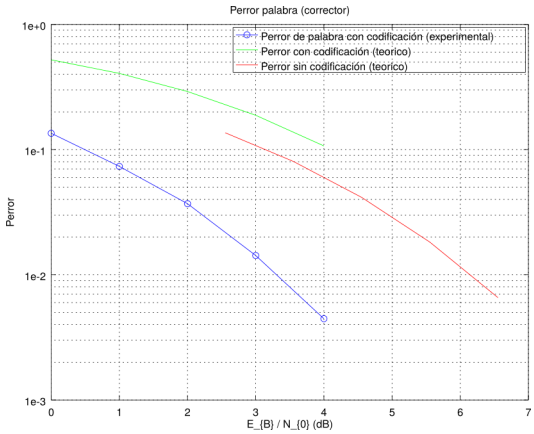

Figura 1: Probabilidad de error de palabra al usar el código como corrector (palabras distintas luego de corregir errores)

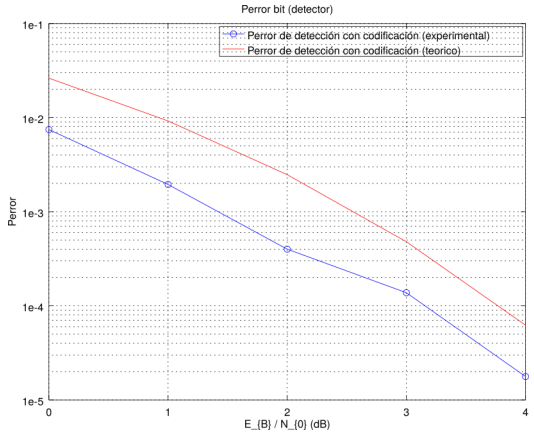

Figura 2: Probabilidad de error de palabra al usar el código como detector (palabras distintas luego de descartar las detectadas con sindrome no nulo)