Métodos Numéricos Avanzados
Un sistema de comunicaciones

María de la Puerta Echeverría (50009), Teresa Fontanella De Santis (52455) y Tomás Mehdi (51014)
Grupo 2 - Instituto Tecnológico de Buenos Aires (I.T.B.A.)

Resumen—En el presente artículo se describe la estimación de un canal de un sistema de comunicación mediante el método de cuadrados mínimos. El entorno de programación que se utiliza es Matlab.

Palabras claves—Sistema de comunicación, estimación de canal, respuesta al impulso, cuadrados mínimos, QR.

1 Introducción

Un sistema de comunicaciones consta, como mínimo, de tres componentes: un transmisor (que envía los datos), un receptor (que los recibe) y el medio por donde se transmiten los datos o *canal*. Dado que éste último modifica la información transmitida, es necesario que el receptor pueda conocerlo, para poder interpretar los datos de forma adecuada. Como generalmente no se conoce, resulta necesario estimarlo. A eso se le llama *estimación del canal*. En el presente informe se estima el canal, utilizando el método QR para cuadrados mínimos.
En la segunda sección se describe la metodología usada y las pruebas realizadas. En la tercera y última sección se muestran los resultados obtenidos y se sacan conclusiones.

2 Metodología utilizada

Para la realización de este trabajo, se toman en cuenta:

2.1 Modelo empleado

El transmisor envía un dato sk cada T segundos, donde s_0 es enviado en $t = 0$, s_1 en $t = T$, s_2 en $t = 2T$, ...
La modificación de los datos por el canal se presenta por la denominada *respuesta al impulso del canal* $\{h_k\}_{k=0}^{L}$, donde L es la longitud de la respuesta al impulso.
Además de ser modificados por el canal, los datos son afectados por ruido blanco Gaussiano aditivo $N_k \sim cN(0, \sigma)$.
Teniendo todo esto en cuenta, cada T segundos el receptor observa:

$$r_n = \sum_{k=0}^{L-1} h_k s_{n-k} + N_n \tag{1}$$

También podemos expresar la Ec. 1 en forma matricial como

$$\vec{r} = H\vec{s} + \vec{N} \tag{2}$$

donde

$$\vec{r} = \begin{pmatrix} r_0 \\ r_1 \\ ... \\ r_M \end{pmatrix}, \vec{s} = \begin{pmatrix} s_0 \\ s_1 \\ ... \\ s_{M-1} \end{pmatrix}, \vec{N} = \begin{pmatrix} N_0 \\ N_1 \\ ... \\ N_{M-1} \end{pmatrix} \tag{3}$$

2.2 Pruebas realizadas

Para efectuar las pruebas correspondientes, se considera utilizar un $L = 1, 10, 30, 50$, aplicando un ruido gaussiano $\sigma = 1$. Se considera este valor por ser lo suficientemente chico para no distorsionar demasiado los datos a transmitir. También se prueba con diferentes longitudes de la cadena de entrenamiento $M = 32, 512, 1024$.
Para analizar cómo se estima en "condiciones óptimas" (verbigracia: sin ruido, etc). también se prueba con $M = 512$ y $\sigma = 0$ (con los valores de L ya mencionados).

3 Resultados y conclusiones

Se realizaron cien corridas con los métodos de búsqueda DFS, BFS, ID, Greedy y Astar para cada uno de los tres tableros con niveles de dificultad baja, media y alta. Se tomó como medida de dificultad del tablero el nivel del juego, cuanto mayor es el nivel del juego, mayor es la dificultad del tablero (es importante aclarar que esta dificultad no tiene relación con la cantidad de movimientos para sacar la ficha azul, sino en como hacer los movimientos para sacarla). Los resultados mostrados son un promedio de esas pruebas. La disposición de las piezas en los tableros puede verse en las imágenes anexas.

Todas las pruebas fueron realizadas con un orden de aplicación de reglas aleatorio. La generación de números

aleatorios en todos los casos fue inicializada con la misma semilla.

La primer batería de pruebas fue realizada sobre el tablero de dificultad baja sin ninguna poda más que la eliminación de ciclos
Con estas primeras pruebas se busca analizar si los resultados obtenidos con los distintos métodos de búsqueda se corresponden con la teoría.

Los resultados de las pruebas pueden verse en la sección anexa.

La segunda batería de pruebas fue realizada con los métodos de poda propios del motor y además se agregó la eliminación de nodos repetidos con el mismo costo. Esto se puede realizar ya que en este problema todas las reglas tienen el mismo costo asociado. Por lo tanto dos nodos repetidos con el mismo costo van a estar siempre en el mismo nivel del árbol y no va a ocurrir que uno pueda ser solución óptima y el otro no.
Con estas segundas pruebas se busca analizar cómo se comportan los distintos métodos de búsqueda para este problema en particular, con las correspondientes mejoras en cuanto a eficiencia, que pueden ser aplicadas debido a las características del mismo.

Los resultados de las pruebas pueden verse en la sección anexa.

Aclaraciones: La fila "nodos visitados sin reiniciar" es un valor relevante sólo para el método ID y cuenta la cantidad de nodos visitados, sin reiniciar el contador en cada iteración. Con algunos métodos en determinadas circunstancias (dificultad del tablero, tipo de heurística, etc.) no se llegaron a completar las cien pruebas debido a tiempo de ejecución o falta de memoria. En estos casos el promedio se calculó sobre el número de pruebas alcanzadas.

4 Conclusiones

De la primer batería de pruebas se puede ver que los datos obtenidos se corresponden con la teoría.
Comparando BFS con ID se ve que ambos llegan a la misma profundidad. Ambos métodos son completos cuando el máximo número de sucesores de un nodo es finito y óptimos cuando el costo no decrece con la profundidad. Estas dos cosas suceden en el presente problema con lo cual ambos métodos son completos y óptimos. Sin embargo se puede ver que ID en el nivel final expande una cantidad de nodos menor que BFS. Además BFS tiene una cantidad mucho mayor de nodos frontera que ID. Esto se debe a que ID va visitando nodos y si no son solución los expande, siempre y cuando no estén en el último nivel de la iteración. Por otro lado BFS expande siempre que visita un nodo que no es solución independientemente del nivel y es por esto que va acumulando más nodos frontera. Esto hace que gaste mayor cantidad de memoria.
Comparando DFS con Greedy se ve que Greedy llega a la solución en una profundidad menor debido a que es un método de búsqueda informado.
Se puede ver también que Astar es el método que llego a la solución óptima expandiendo la menor cantidad de nodos. Comparando las heurísticas en este método se observa que la segunda al ser admisible expande menor cantidad de nodos que la primera ya que encuentra antes la solución.

De la segunda batería de pruebas se puede ver que la poda beneficia a todos los métodos de búsqueda. El que más se beneficia es BFS en el que se ve que por ejemplo la cantidad de nodos frontera disminuyó considerablemente. Esto se debe a que expande una cantidad de nodos mucho menor porque no visita nodos repetidos.
A medida que se aumenta la dificultad del tablero se ve que algunos métodos de búsqueda comienzan a fallar en algunas situaciones. Por ejemplo en el tablero de dificultad alta ni DFS ni Greedy pueden completar todas las iteraciones. En este tablero también se puede ver que a este nivel de dificultad Astar comienza a ser el método más óptimo en cuanto a cantidad de nodos expandidos y nodos frontera (también en cuanto al tiempo de ejecución a pesar de que no sea un parámetro de comparación tan preciso como el resto) y se separa claramente de los demás. Además se puede observar que la heurística admisible da resultados mucho mejores que la no admisible.

References

[1] John Mattews and Kurtis Fink, *Métodos Numéricos con Matlab*

Pruebas sin poda

Pruebas tablero dificultad baja

Estrategia	DFS	BFS	ID	Greedy (h1)	Greedy (h2)(*1)	Astar (h1)	Astar (h2)
Profundidad	128	4	4	121	69	4	4
Nodos expandidos	130	4091	3382	121	69	3289	2030
Nodos expandidos sin reiniciar	130	4091	4355	121	69	3289	2030
Nodos frontera	815	31985	22	776	529	25764	18017
Total estados	947	36077	3405	898	599	29055	20048
Tiempo (ms)	14	369	47	14	7	2415	1434

*(*1) Promedio de 4 corridas*

Pruebas con poda

Pruebas tablero dificultad baja

Estrategia	DFS	BFS	ID	Greedy (h1)	Greedy (h2)	Astar (h1)	Astar (h2)
Profundidad	66	4	4	66	49	4	4
Nodos expandidos	125	256	313	97	346	221	192
Nodos expandidos sin reiniciar	125	256	558	97	346	221	192
Nodos frontera	264	155	12	266	234	172	266
Total estados	391	412	327	365	582	394	460
Tiempo (ms)	27	38	27	18	144	33	33

Pruebas tablero dificultad media

Estrategia	DFS	BFS	ID	Greedy (h1)	Greedy (h2)	Astar (h1)	Astar (h2)
Profundidad	211	14	14	202	174	14	14
Nodos expandidos	976	1215	5371	727	588	1188	1283
Nodos expandidos sin reiniciar	976	1215	22528	727	588	1188	1283
Nodos frontera	702	60	23	671	586	73	179
Total estados	1679	1276	5395	1399	1175	1262	1463
Tiempo (ms)	698	297	7161	405	441	297	345

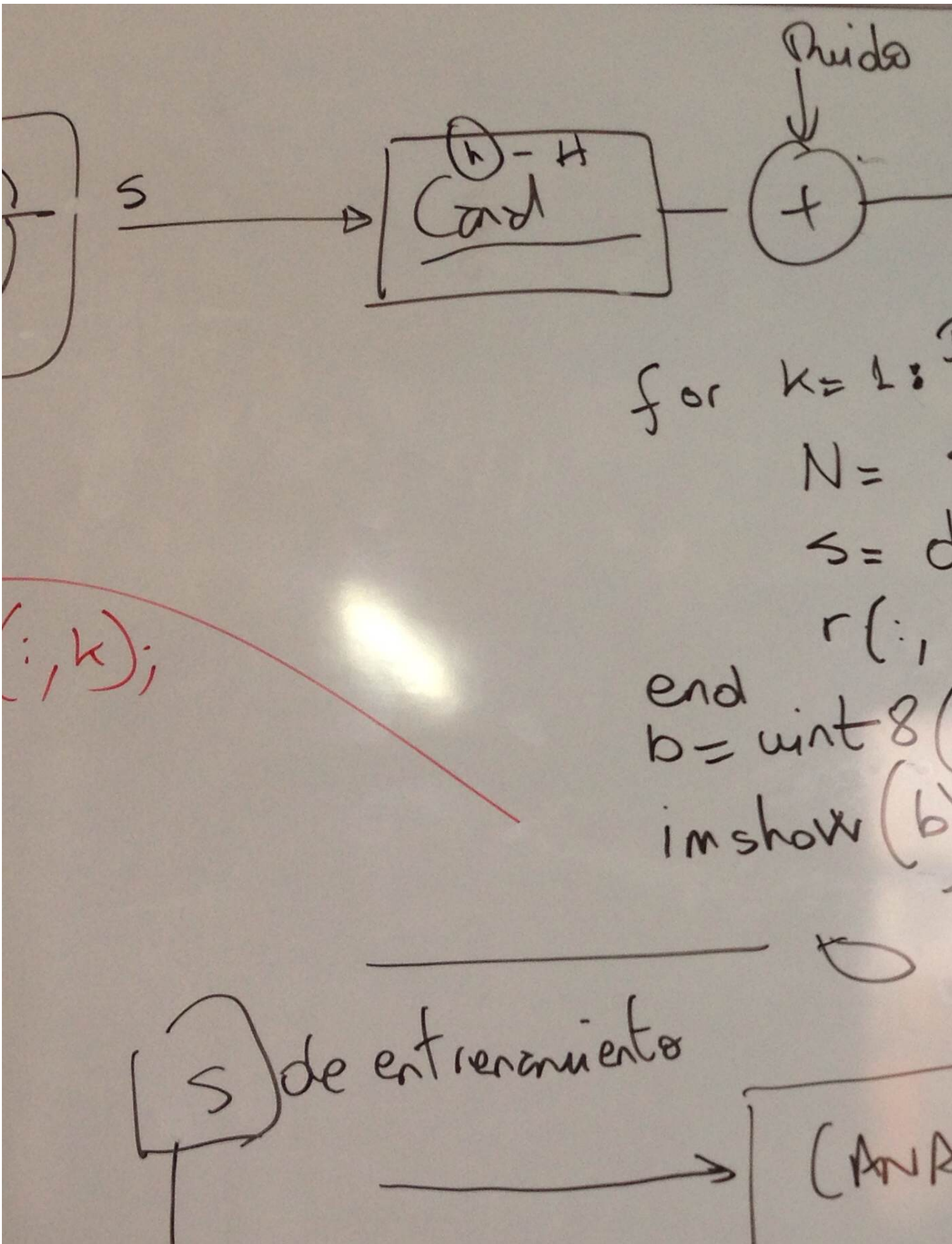
Pruebas tablero dificultad alta

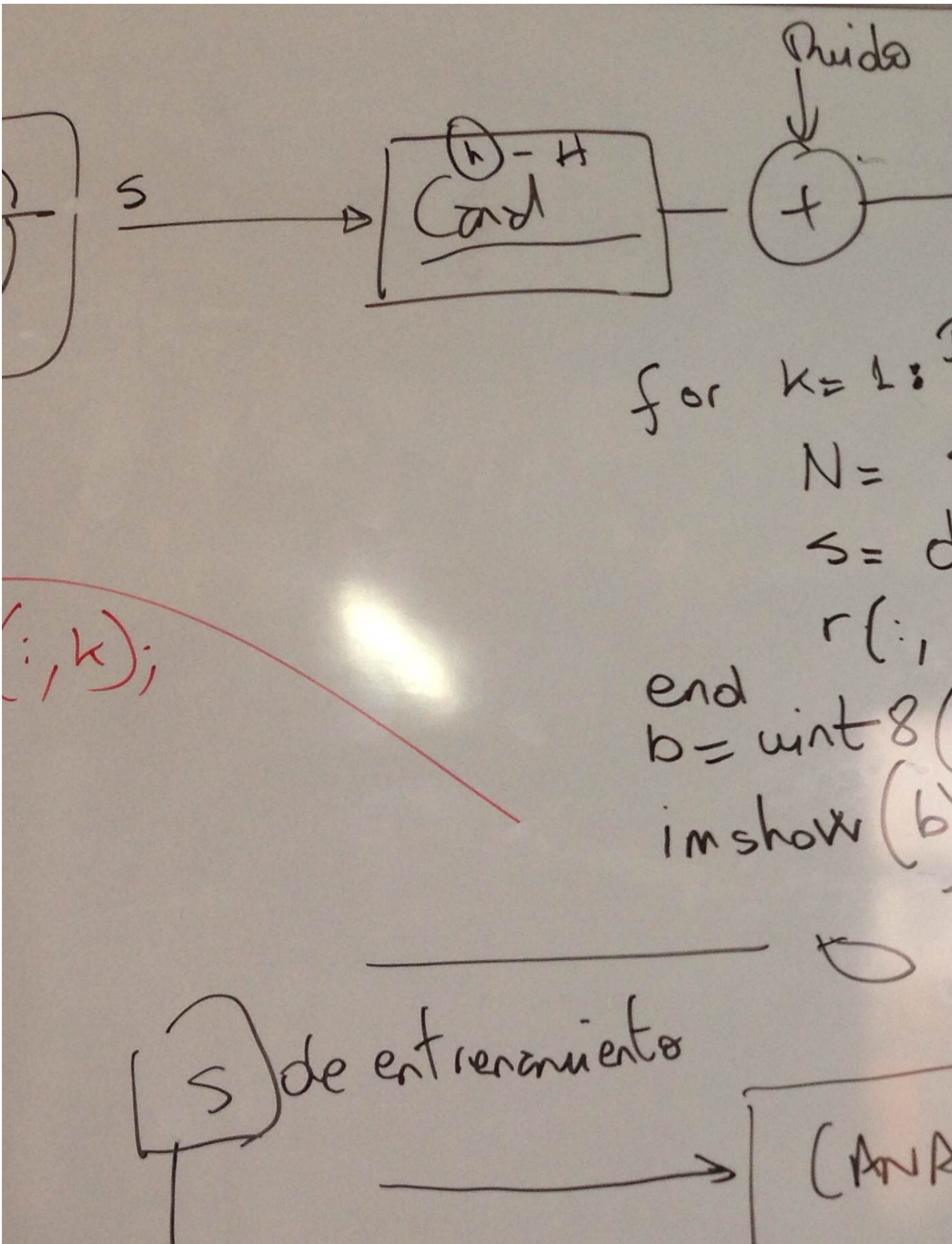
Estrategia	DFS(*1)	BFS	ID	Greedy (h1)(*2)	Greedy (h2)(*3)	Astar (h1)	Astar (h2)
Profundidad	293	10	10	105	57	10	10
Nodos expandidos	312	1605	3360	111	57	1430	784
Nodos expandidos sin reiniciar	312	1605	9596	111	57	1430	784
Nodos frontera	1249	716	22	389	214	737	539
Total estados	1562	2323	3384	501	272	2169	1325
Tiempo (ms)	90	491	1847	20	77	426	200

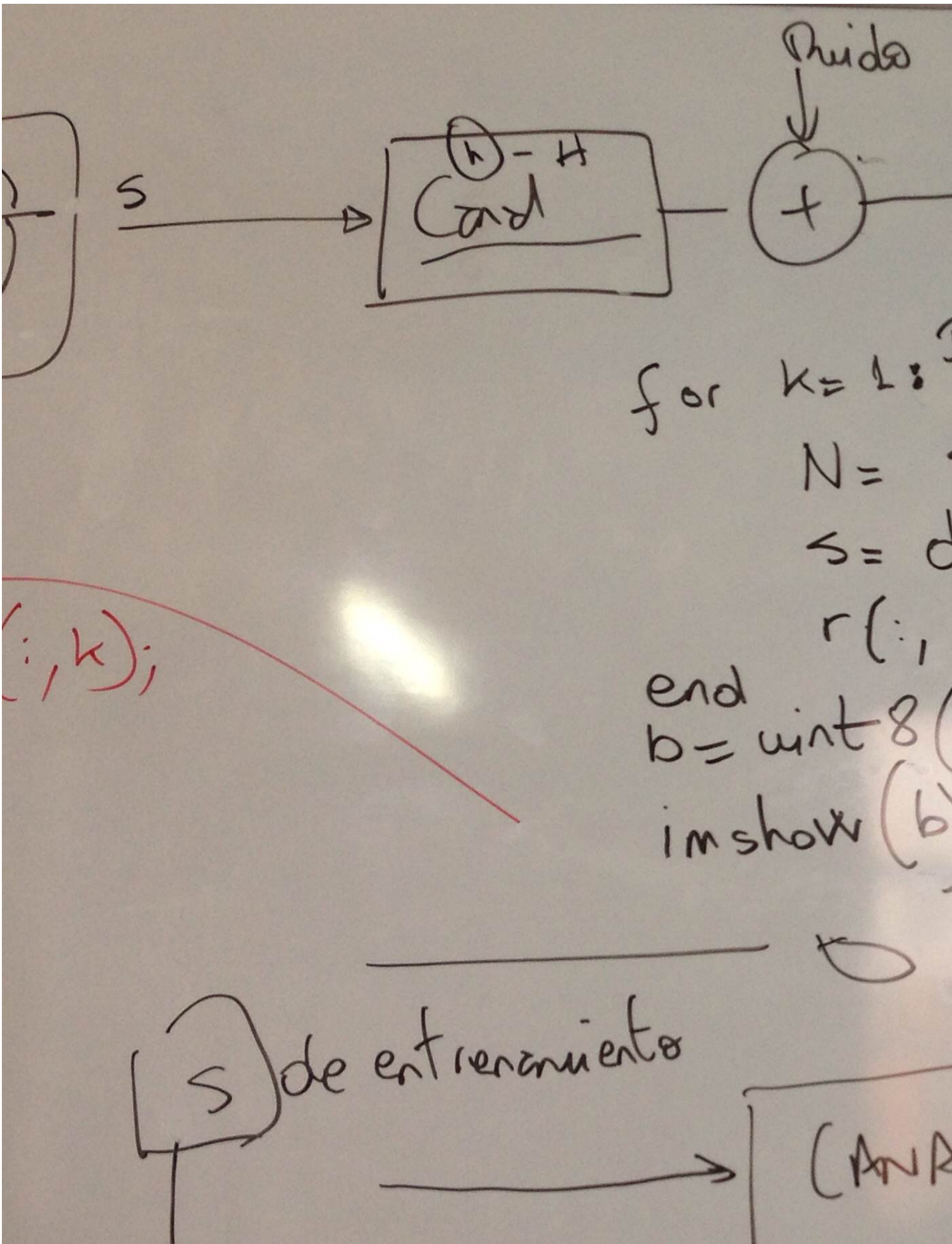
*(*1) Promedio de 11 corridas*

*(*2) Promedio de 7 corridas*

*(*3) Una corrida*


Ruido
s
h - H
Conv
+
for k=1:
N=
s=
r(:,
end
b=uint8
imshow(b)
(:,k);
s de entrenamiento
(ANA


Ruido
h - H
Conv
s
+
for k=1:
N=
s=
r(:,
end
b=uint8
imshow(b)
(:,k);
s de entrenamiento
(ANA


Ruido
s
h - H
Conv
+
for k=1:
N=
s=
r(:,
end
b=uint8
imshow(b)
(:,k);
s
de entrenamiento
(ANA