

Un sistema de comunicaciones

Julián Gutiérrez (51141) — Pablo Pauli (51185) — Ivan Itzcovich (53891) — Gustavo Del Giudice (51289)

Resumen

Un sistema de comunicaciones establece un medio por el cual se envía y se recibe un mensaje a través de un canal. El usuario de este sistema de comunicaciones espera, aún desconociendo el mecanismo de transmisión que el mensaje transmitido sea exactamente idéntico al que lo recibe. Para esto, el canal debe mantener la identidad del mensaje y conservar su estructura, de otra manera el mensaje se vería modificado, y en consecuencia, el receptor estaría adquiriendo un mensaje distinto al original. Esta situación es imposible de lograr, puesto que un canal, al ser un medio físico, impone ciertas modificaciones al mensaje que está viajando por él. Estas modificaciones se deben a un comportamiento propio del canal, como por ejemplo la resistencia de los materiales del canal o la irrupción del medio en el cual habita el canal, el cual es imposible ignorar. En particular, en este trabajo se analiza el caso de un canal de índole informático, en el cual viajan datos digitales por un sistema de comunicación discreto en banda base.

El objetivo principal es poder desarrollar un sistema que, a partir de un mensaje recibido, se pueda revertir el problema que ocasionarán las modificaciones que le imprime el canal; para luego, una vez que se conoce dicho comportamiento, poder reinterpretar un mensaje recibido para obtener el mensaje original que fue enviado.

1. Introducción

Un modelo muy simple de un sistema de comunicaciones consiste de un transmisor que envía un dato cada cierto tiempo. Los datos son modificados por el canal, es decir, por el medio donde son transmitidos. Esta modificación se genera por la respuesta al impulso del canal. Además de ser modificados por el canal, los datos son afectados por ruido blanco Gaussiano aditivo. Teniendo en cuenta estas modificaciones, el receptor obtiene una respuesta.

Lo que nos interesa es, usando la respuesta que obtiene el receptor, conseguir la señal que se envió al inicio de la comunicación. El problema es que las modificaciones al canal y la longitud de la respuesta al impulso no suelen ser conocidas, por lo tanto debemos estimarlos. Si la longitud es conocida, es posible estimar la respuesta al impulso del canal mediante cuadrados mínimos. Para esto, hay que enviar una señal conocida por el receptor. La denominamos secuencia de entrenamiento. Luego, se estima el canal planteando otro problema de cuadrados mínimos.

A continuación, se va a detallar la metodología con la que se realizó el trabajo, para luego exponer los resultados que se obtuvieron al realizar las distintas pruebas que requería el mismo, seguido de las conclusiones obtenidas.

2. Metodología

2.1. Modelado

El mensaje se transmite por un transmisor, el cual envía un dato del mensaje cada cierto intervalo de tiempo. Si el mensaje es s_k , entonces s_0 es enviado en $t=0$, s_0 en $t=T$, s_2 en $t=2T$, etc. Cada dato transmitido es modificado por el canal como respuesta al impulso que genera el mismo. Esta modificación se puede presentar

como h_k (con $k=0$ a L), donde L es la longitud de la respuesta al impulso. Además, los datos son perturbados con ruido blanco Gaussiano aditivo, $N_k \sim \mathcal{N}(m, \sigma)$.

La ecuación del dato recibido responde entonces a la siguiente:

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

Esta ecuación se expresa de forma matricial como:

$$\vec{r} = H\vec{s} + \vec{N} \quad (2)$$

Donde r es el mensaje recibido, compuesto por los datos recibidos; s es el mensaje transmitido, N es el ruido blanco Gaussiano, y H es la matriz de toeplitz, que corresponde a las modificaciones que se establecen en el mensaje transmitido, de tamaño $M \times M$ (siendo M la cantidad de datos que posee el mensaje a transmitir).

La ecuación (2) se puede expresar también de la siguiente manera:

$$\vec{r} = S\vec{h} + \vec{N} \quad (3)$$

Manteniendo r y s equivalentes, siendo S la matriz toeplitz del dato transmitido y h el vector de la perturbación del impulso del canal.

2.2. Inicializando y Perturbando

En primer lugar, fue necesario simular el vector h , es decir, la respuesta al impulso del canal que se genera el transmitir el mensaje. Para esto, se consideró $L=5$, y luego el h se lo obtuvo a través de una simulación pseudo-aleatoria con distribución normal. La variable σ , fue inicializada en 0,01. Para obtener H , matriz de toeplitz que luego se va a utilizar para simular la transmisión, se utilizó la función “toeplitz” de Octave. Esto se puede observar en el código de “initializeCannal.m”

del anexo. Vale destacar la ya mencionada matriz H cumple la siguiente propiedad:

$$\forall \quad a_{i,j} \in H \rightarrow a_{i,j} = a_{i+1,j+1} \quad (4)$$

Con el h establecido, que de ahora en adelante va a ser referido como h “real”, se puede simular la transmisión de un mensaje.

El objetivo principal del trabajo es el de “revertir el problema que ocasionan las modificaciones que le imprime el canal”, tal como dice el Resumen. Para realizarlo, se tomo como mensaje a perturbar (y luego a reinterpretar), la famosa imagen de “Lena”:



Figura 1: Imagen 512

Esta imagen sirve como imagen de prueba para los algoritmos de compresión de imagen, se lo considera un estándar científico. La imagen se transmite por filas por conveniencia, es decir, que nuestro dato s_k es la fila k de la imagen. Para la transmisión, se usa la ecuación (2) del Modelado, por cada fila que se transmite. Es decir, fue necesario realizar un loop iterando sobre todas las filas de la imagen para perturbarlas. Dicho proceso se incluye en el ciclo “for” de “simulation.m” del anexo. Es importante mencionar que en este mismo ciclo se realiza la tarea de reinterpretación, que se explica más adelante. En la sección Resultados, se puede apreciar las modificaciones que sufre “Lena” al pasar por el canal de transmisión. También es importante aclarar que se realizaron múltiples recuperaciones de imágenes, variando ciertos parámetros que más adelante se detallarán, y para cada una de estas se simuló la transmisión, obteniéndose así múltiples imágenes recibidas.

Otra aclaración importante, visible claramente en el código, que es importante resaltar es que por un tema de eficiencia, para disminuir la cantidad de operaciones que debe realizar el Octave para operar con los vectores, cada fila que se envía (y luego se reinterpreta) es transmitida por partes, exactamente en 16 (esta modificación permitió que el tiempo que lleva cada iteración por fila bajara notablemente).

2.3. Estimando h

Para lograr el objetivo de revertir las modificaciones que sufre el mensaje, es necesario conocer h , pues es indispensable para reconstruirlo. Ahora bien, el h

“real” es desconocido, el canal funciona como una “caja negra” sobre el cual el usuario del sistema de comunicación sólo puede conocer la entrada y la salida. Por la ecuación (2), lo que se conoce es S y r . Lo que se quiere estimar es h , entonces, se puede plantear un problema de cuadrados mínimos, obteniendo el h que minimice la siguiente ecuación:

$$\|S\vec{h} - \vec{r}\|_2^2 \quad (5)$$

Siendo h que minimiza la solución del siguiente sistema, denominadas “ecuaciones normales”:

$$S^t \cdot S \cdot \vec{h} = S^t \cdot \vec{r} \quad (6)$$

Este h constituye la estimación del h “real”.

El procedimiento para estimar h fue el siguiente. Se tomó como s una cadena random, de longitud E , el cual constituye el mensaje “conocido” que se posee para el proceso. Al igual que antes, S se obtiene invocando la función `toeplitz` de *Octave*. Esta matriz S fue sometida a la secuencia de “entrenamiento”. Esto quiere decir que S fue enviada por el canal, por la “caja negra”, en donde sufrió las modificaciones por respuesta al impulso del canal y donde fue perturbado por el ruido blanco Gaussiano. Una vez invocada la secuencia de entrenamiento (función “train.m” del anexo), se obtuvo r .

Con S y r se procedió a solucionar el problema de cuadrados mínimos. Para ello, se recurrió a la técnica de Cholesky. Aplicando esta técnica, es posible representar una matriz simétrica definida positiva como el producto de una matriz G (triangular superior) con G transpuesta (triangular inferior). Es simple demostrar que $S^t \cdot S$, presente en la ecuación (5), es una matriz simétrica definida positiva (la demostración excede al objetivo de este informe). Así, la ecuación normal con el reemplazo de G y G^t , resulta en:

$$G \cdot G^t \cdot \vec{h} = S^t \cdot \vec{r} \quad (7)$$

Realizando la siguiente sustitución, se facilita hallar la solución de las ecuaciones normales, pues la naturaleza de las matrices que se utilizar permiten hacer una sustitución regresiva y luego una progresiva.

$$G^t \cdot \vec{h} = \vec{w} \quad (8)$$

Sustitución regresiva.

$$G \cdot \vec{w} = S^t \cdot \vec{r} \quad (9)$$

Sustitución progresiva.

Esta rutina, se encuentran en “cholesky.m” y “back-Sustitucion.m” del anexo.

La estimación de h , que hace uso de todas las funciones mencionadas, se encuentra en la función “estimah.m” del anexo. Ejecutando la función “calculatError.m”, se calculó el error de estimación.

Este procedimiento de estimar h fue realizado variando los parámetros de E, de sigma, y de L. Los resultados del error en la estimación de los distintos h calculados, se encuentran en la sección de Resultados.

2.4. Recuperando el mensaje

Una vez que se tiene la estimación de h, se obtiene de la misma manera que antes la matriz toeplitz H. De la ecuación (2), se puede obtener un s que minimice la función:

$$\|H\vec{s} - \vec{r}\|_2^2 \quad (10)$$

Este s es solución de las .“ecuaciones normales”:

$$H^t.H.\vec{s} = S^t.\vec{r} \quad (11)$$

Al igual que en el proceso para estimar h, nuevamente se resolvió el problema de cuadrados mínimos con la técnica de Cholesky, pero en este caso, se obtiene la reinterpretación(s) del mensaje recibido r. Es importante considerar que el error en la imagen se reconstruye es evidente, teniendo en cuenta que se usa una estimación del h “real” y que no es posible contrarrestar el efecto del ruido blanco Gaussiano.

Como se anticipó antes, en el mismo ciclo en el que las filas de “Lena” se perturban, las mismas se recomponen. Es decir, al igual que en la transmisión, la recomposición de la imagen se realiza por filas. Esto se puede observar en el ciclo “for” de “simulation.h” del anexo.

3. Resultados

Las pruebas se corrieron modificando tres parámetros, la longitud de la respuesta al impulso (L), la longitud de la cadena que se sometió a la secuencia de .“entrenamiento” para estimar h (E), y la desviación del random de distribución normal que establece el ruido blanco Gaussiano (sigma).

Los valores que tomó L fueron 1,3,5. Los de E fueron 512,32,1024. Y sigma tomó los valores de 0.01,0,0.1.

Es importante destacar que en las imágenes se observan 16 rayas verticales. Esta característica es consecuencia de transmitir las filas en 16 partes para agilizar el algoritmo.

$$E = 512, \sigma = 0.01$$

Suponiendo L=1.

El error al estimar h fue:

$$\begin{aligned} 3.14128940206992e-07 \\ 1.01778682854363e-01 \\ 5.34726605916866e-02 \\ 3.83644165904570e-02 \end{aligned}$$

1.20454794364806e-01

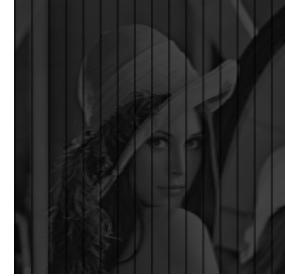


Figura 2: Imagen Recibida



Figura 3: Imagen Recuperada

Suponiendo L=3.

El error al estimar h fue:

$$\begin{aligned} 2.21210531670124e-06 \\ 2.23888955847018e-06 \\ 2.46672604342635e-07 \\ 1.50809537170849e-01 \\ 1.70955645505803e-01 \end{aligned}$$

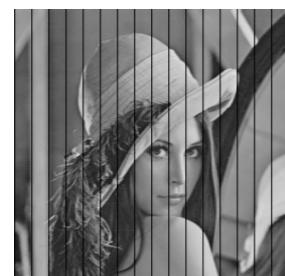
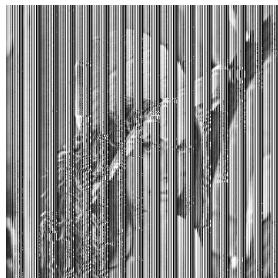


Figura 4: Imagen Recibida



7.29150556624208e-02

Figura 5: Imagen Recuperada

Suponiendo L=5.

El error al estimar h fue:

2.82588050182220e-07
1.32275305559093e-05
1.96310588279625e-06
4.62617064481835e-06
5.62121598386700e-07

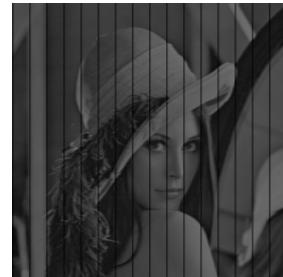


Figura 8: Imagen Recibida

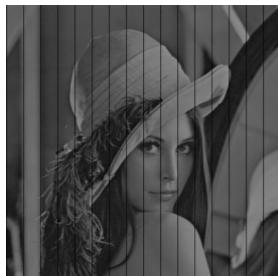


Figura 6: Imagen Recibida



Figura 9: Imagen Recuperada

Suponiendo L=3.

El error al estimar h fue:

1.97271398003074e-06
1.56387741185871e-05
1.05949822686788e-05
8.36945405055923e-02
3.36241020002239e-02



Figura 7: Imagen Recuperada

E = 32, $\sigma = 0.01$

Suponiendo L=1.

El error al estimar h fue:

5.57204903386954e-06
8.94206341063025e-02
6.96736749314932e-03
1.31128977056077e-01

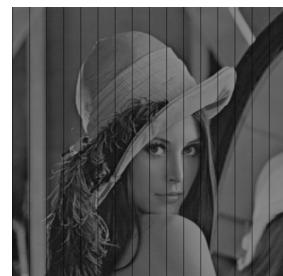
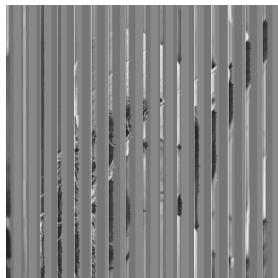


Figura 10: Imagen Recibida



1.54787518630523e-03

Figura 11: Imagen Recuperada

Suponiendo L=5.

El error al estimar h fue:

6.09505131402011e-06
2.79206028110310e-05
3.93434730151798e-05
3.38890748193821e-05
2.85690816251571e-05

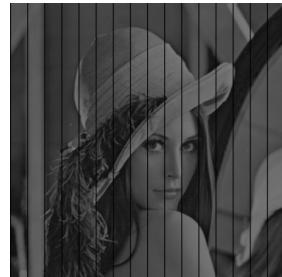


Figura 14: Imagen Recibida

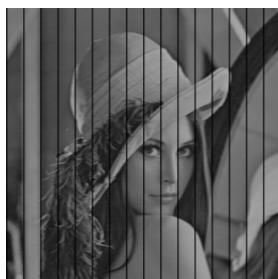


Figura 12: Imagen Recibida

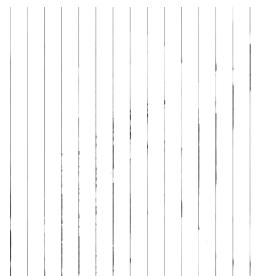


Figura 15: Imagen Recuperada

Suponiendo L=3.

El error al estimar h fue:

3.89915389698015e-07
2.04317964785927e-06
1.39987199031244e-06
1.58869179782212e-01
2.67648902523867e-03



Figura 13: Imagen Recuperada

E = 1024, $\sigma = 0.01$

Suponiendo L=1.

El error al estimar h fue:

7.41564685047269e-07
1.22306742447753e-01
2.84237281719386e-01
6.17325936170009e-02

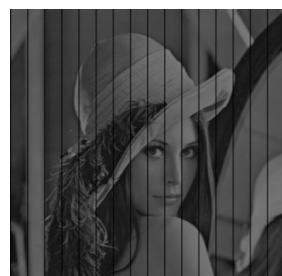
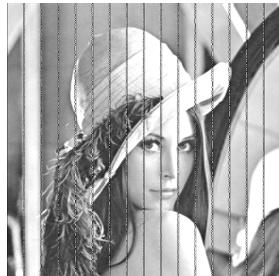


Figura 16: Imagen Recibida



1.40969371525189e-01

Figura 17: Imagen Recuperada

Suponiendo L=5.

El error al estimar h fue:

8.57149750378705e-07
5.07209018510424e-06
2.00696787977517e-06
3.09745779877857e-06
6.08170666729912e-07



Figura 20: Imagen Recibida

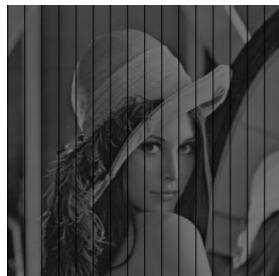


Figura 18: Imagen Recibida

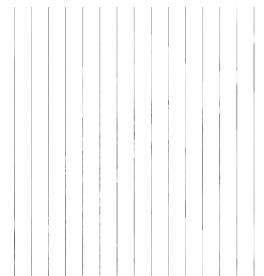


Figura 21: Imagen Recuperada

Suponiendo L=3.

El error al estimar h fue:

9.85322934354826e-16
1.38777878078145e-17
9.71445146547012e-16
6.02180470483583e-02
2.54199365682852e-01



Figura 19: Imagen Recuperada

E = 512, σ = 0

Suponiendo L=1.

El error al estimar h fue:

6.93889390390723e-18
1.87016036254111e-01
1.05858977334859e-01
2.67652018807568e-01

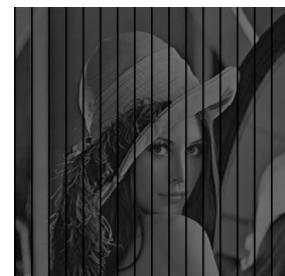


Figura 22: Imagen Recibida

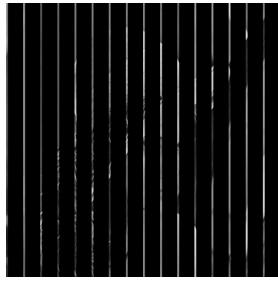


Figura 23: Imagen Recuperada

Suponiendo L=5.

El error al estimar h fue:

1.20736753927986e-15
1.87350135405495e-16
2.29330443524134e-15
3.19189119579733e-16
8.88178419700125e-16



Figura 24: Imagen Recibida



Figura 25: Imagen Recuperada

E = 512, $\sigma = 0.1$

Suponiendo L=5.

El error al estimar h fue:

2.77818408822572e-06
6.39480579725515e-07
8.56564309231755e-07
1.47781988831669e-05

4.98113753122191e-06

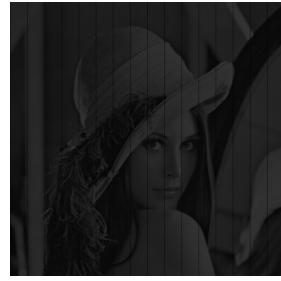


Figura 26: Imagen Recibida



Figura 27: Imagen Recuperada

4. Conclusiones

Como se mencionó anteriormente, las imágenes recibidas nunca permanecen iguales a las que fueron enviadas, puesto que se reinterpreta el mensaje con una estimación de la respuesta al impulso del canal real y además, es imposible contrarrestar el efecto del ruido blanco Gaussiano. Sin embargo, modificando ciertos parámetros en el proceso de reinterpretación del mensaje, en particular E, L, y sigma; es posible obtener una peor o una mejor aproximación al mensaje original transmitido. Es evidente que cuanto menor sea el error de estimación, menos se va a ver alterada la imagen reconstruida.

El primer parámetro a analizar es L, es decir, la longitud de la respuesta al impulso. Como el L real es 5, es lógico que al suponer un Lguess menor, se esté estimando un h menos exacto. Esto se debe que al realizar las pruebas con un Lguess menor a 5, se estiman los primeros "Lguess" valores de h real, por lo que el error se mantiene en un valor esperado en estos valores de h (orden menor a -6), pero los h que no fueron estimados, permanecen en 0. Esto provoca que al recuperar la imagen, es-

tos h que no fueron estimados interfieran en la correcta reinterpretación de la imagen. En consecuencia, como se puede observar en los Resultados, a medida que el L_{guess} que se supone se acerca al L real, el error disminuye y la semejanza entre la imagen recibida y la recuperada es superior. Es más, suponiendo un $L=x$, si se observa el vector del error en la estimación, los primeros x valores son coherentes con un error, mientras los demás no.

El parámetro E tiene que ver con la longitud del mensaje conocido que se usa para estimar h , es decir, el mensaje que se somete a la secuencia de “entrenamiento”. Tal como se nota en los resultados, cuanto más se “entrena” la cadena, el error es menor, y así la estimación del h es mejor. Analizando en un mismo L distintos E , se puede observar como mejora el error

cuando E es más grande.

Por último, σ . Este parámetro hace referencia a la desviación típica del random con distribución normal que establece el ruido blanco Gaussiano. Notar que cuando σ es 0, el error en los valores de h estimados bajan al orden de -16, es decir, sin ruido, la estimación es casi exacta. Evidentemente, si la perturbación por el ruido blanco Gaussiano es menor, la tarea de reinterpretación se facilita, pues, como se dijo antes, esta perturbación es imposible de tener en cuenta en el proceso para recuperar el mensaje.

Se puede apreciar que la mejor aproximación y la mejor recuperación de la imagen se obtiene cuando se reúnen casi todos los requisitos óptimos de los tres parámetros (a pesar del E que podría ser 1024), $L=5$, $E=512$ y $\sigma=0$.

5. Anexo

Aquí se pueden ver las funciones de *GNU Octave* utilizadas para este análisis.

El *script initializeCannal.m* inicializa las variables del canal.

```
initializeCannal.m

%Simulates the initialization of the communications cannal
%params: M columns of the image to be sent.
function initializeCannal(M)
global h;
global L;
global H;
global sigma;
sigma = 0.01;
L = 5;
ganancia = 1/10;
h = ganancia * (1+randn(L,1));%this h is unknown, we want to estimate it.
H = toeplitz([h.' zeros(1,M-L)],zeros(1,M));
endfunction
```

El *script simulation.m* implementa la simulacion del sistema de comunicacion .

```

simulation.m

%Simulates the whole canal transmition problem.
%Note that the h used here is not global, we want to find it.
function simulation()
global L;

format long;

original = imread('../img/lena512.bmp');
a = double(original);
E = 512;
chunk_amount = 16;
M = size(a,2) / chunk_amount;

initializeCannal(M);
%% First Part: Estimate h given a sequence of known bytes %%
h = estimateh(E,M);
e = calculatehError(h)
%% Second Part: transmit image and retrieve it with the h we estimated %%
global Lguess;
H = toeplitz([h.' zeros(1,M-Lguess)],zeros(1,M));
P = size(a,1);
r = zeros(M,P);
s = zeros(M,P);
G = cholesky(H' * H);

for k=1:rows(a)
k
fflush(stdout);
for l = 1:chunk_amount
ii = (l-1)*M+1:l*M;
r(k,ii) = transmit(a(k,ii)');
s(k,ii) = backSustitution(H,r(k,ii)',G)';
endfor
endfor
r = uint8(r);
s = uint8(s);
saveImages(r, s);
figure(1);
imshow(original);
title('Original image');
figure(2);
imshow(r);
title('Received image');
figure(3);
imshow(s);
title('Corrected image');
endfunction

```

El script `train.m` implementa la secuencia de entrenamiento.

```
train.m
```

```
%This function simulates the passage  
%of the train sequence through the channel.  
%params: S the toeplitz matrix containing the train sequence.  
%observations: it forces the passage of S  
%by taking less elements of h(unknown) if it has to.  
%return: r what you would receive on the other end of the channel.  
function r = train(S)  
global h;  
global sigma;  
r = S * h(1: size(S,2));  
r = r + sigma*randn(size(r));  
endfunction
```

El script `transmit.m` simula la transmicion por el canal.

```
transmit.m
```

```
%%%%%  
%Simulates the transmission of sequence s through a canal.  
%params: s, column vector. The sequence.  
%return: r, column vector. What you receive.  
function r = transmit(s)  
global H;  
global L;  
global sigma;  
r = H*s;  
r = r + sigma*randn(size(r));  
endfunction
```

El script `cholesky.m` obtiene la descomposición de Cholesky de una matriz simétrica.

```
cholesky.m
```

```
%%%  
%Decomposes matrix A into matrix G with Cholesky's decomposition.  
%it assumes that A is definite positive (min(eig(A)) <= 0)  
function G = cholesky(A)  
G = zeros(size(A));  
  
for j = 1:columns(A)  
G(j, j) = sqrt(A(j,j) - sum(G(j, 1:(j-1)).^2));  
  
for i = (j+1):rows(A)  
G(i, j) = (A(i, j) - sum(G(i, 1:(j-1)) .* G(j, 1:(j-1)))) / G(j,j);  
endfor  
endfor  
endfunction
```

El script `minimumSquares.m` resuelve los cuadrados mínimos por Cholesky.

```

minimumSquares.m

function x = minimumSquares(A, y, G)
%load cholesky.m;

%G = cholesky(A' * A);
%w = zeros(columns(A), 1);
%x = zeros(columns(A), 1);

%for i=1:length(w)
% w(i) = ((A' * y)(i) - G(i, 1:(i - 1)) * w(1:(i-1))) / G(i, i);
%endfor

%l = length(x);

%for i=l:-1:1
% x(i) = (w(i) - G'(i, i+1:l) * x(i+1:l)) / G'(i, i);
%endfor

w = inv(G) * A' * y;
x = inv(G') * w;
endfunction

```

El script `backSustitution.m` hace la sustitucion hacia atras.

```

backSustitution.m

%Solves the backsustitution problem needed to
%complete minimum squares with Cholesky's decomposition
%G*w = A'*y
%G'*x = w
%parms: A Matrix
%parms: G Cholesky Matrix
%prams: y Column vector
%return: x Column vector
function x = backSustitution(A, y, G)
n = columns(A);
w = zeros(n, 1);
x = zeros(n, 1);
temp = (A' * y);%for eficciency
%We treat the scalar case in a separate way
%because of how octave handles matrixes.
if(max(size(temp))==1)
w = temp/G;
x = w/G;
else
for i=1:n
w(i) = (temp(i) - G(i, 1:(i - 1)) * w(1:(i - 1))) / G(i, i);
endfor

for i=n:-1:1
x(i) = (w(i) - G'(i, i+1:n) * x(i+1:n)) / G'(i, i);
endfor
endif
endfunction

```

El script `saveImages.m` guarda las imagenes.

```
saveImages.m
```

```
function ans = saveImages(noisy, corrected)
imwrite(uint8(noisy), gray(256), '../img/received.png');
imwrite(uint8(corrected), gray(256), '../img/corrected.png');
endfunction
```

El script `calculatehError.m` calcula el error de h.

```
calculatehError.m
```

```
%Calculates the error between h_estimated and the
%h of the cannal.
function e = calculatehError(h_estimated)
global h;
e = abs(h - [h_estimated; zeros(length(h)-length(h_estimated),1)]);
endfunction
```

El script `estimateh.m` estima el valor de h.

```
estimateh.m
```

```
%Estimates h of the cannal.
%params: E the lenght of the train sequence.
%params: M columns of the image to be sent.
function h = estimateh(E,M)
global L;
global Lguess;
Lguess = 5;
MAX_IMGVALUE = 255;
trainSequence = MAX_IMGVALUE*rand(1,E);
S = toeplitz([trainSequence zeros(1,M-E)], zeros(1,Lguess));
r = train(S);
G = cholesky(S' * S);
h = backSustitution(S,r,G);
endfunction
```