INSTITUTO TECNOLÓGICO DE BUENOS AIRES

# Trabajo Práctico $N^o 1$

# Métodos Numéricos

Grupo 1:
Mestanza Joaquín
Müller Malena
Nowik Ariel
Regueira Marcelo
Reller Thomas

Profesor:
Fierens Pablo
Álvarez Adrián Omar

April 11, 2019

# 1 INTRODUCCIÓN

Toda señal enviada por un sistema de comunicación sufre perturbaciones durante el proceso de transmisión, y es por eso que se desea reducir el error de la información recibida. Un modelo discreto sencillo de un sistema de comunicaciones es el siguiente. Periódicamente, cada T segundos el transmisor envía un dato $s_k$ considerando como instante inicial a $t = 0$. Para los objetivos de este trabajo, se clasifican dos tipos de errores de distinta naturaleza asociados al proceso de transmisión: error sistemático de respuesta impulsiva del canal (el cual depende del medio donde la información se propaga y de la señal), y ruido blanco gaussiano que agrega incertezas que no podemos mejorar. El objetivo entonces es el de recuperar la señal original, teniendo que:

1. cuantificar los errores vinculados,
2. dar un estimador que se aproxime lo más que se pueda a la señal original,
3. y resolver sistemas de ecuaciones lineales eficientemente para así evitar errores numéricos (como redondeo, esto último se debe a que la computadora tiene una representación finita de números reales).

Modelamos el problema como sigue (para sistemas LTI): La información es modificada por el canal a través de su "respuesta impulsiva". Esto es la respuesta del sistema (en este caso, el canal) frente a una señal de entrada en particular que se conoce como impulso unitario ó delta de dirac. Matemáticamente, esto permite expresar la salida de un sistema en general como la convolución de su respuesta impulsiva con la señal de entrada. A su vez, la señal transmitida es afectada por ruido blanco Gaussiano aditivo , donde $N_k \sim cN(0,\sigma)$. Entonces, periódicamente la información recibida es

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

Donde $h$ es la respuesta impulsiva previamente mencionada. Matricialmente, esto se puede expresar de dos formas distintas:

$$\vec{r} = H\vec{s} + \vec{N} \tag{1.1}$$

$$\vec{r} = S\vec{h} + \vec{N} \tag{1.2}$$

Siendo en todos los casos

- **h** referente a información del canal
- **s** referente a información de la señal original
- **n** referente al ruido agregado a la señal
- **r** referente a la señal recibida

Además, puede observarse que como la señal recibida es una variable aleatoria, dado que el ruido gaussiano es aleatorio y no hay manera de conocer la respuesta impulsiva con precisión infinita; debe elegirse adecuadamente el estimador de $S_k$ de manera tal de minimizar el error – cuantificándose por el método de cuadrados mínimos de estadística.

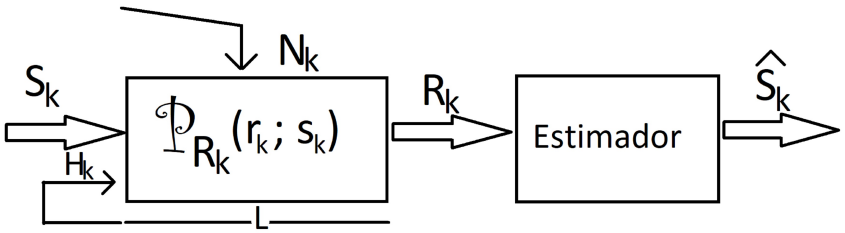



Figure 1.1: Representación del sistema y la estimación de la señal de entrada.

Como se puede ver en la Figura 1.1, se recibe una señal $R_k$ como respuesta a una señal enviada $S_k$. Como el transmisor tiene errores de dos tipos, además del correspondiente estimador es necesario acotar el error.

## 2 Estimación de la señal enviada

Es necesario hacer una estimación de dos cosas distintas. Por un lado se debe estimar el canal ($\vec{h}$) y por otro lado, la imagen original ($\vec{s}$).

### 2.1 Estimación del canal de comunicación

Primero se hace una estimación de $\vec{h}$ con el método de cuadrados mínimos, empleando la descomposición de Cholesky, como se explica a continuación. Para averiguar $\vec{h}$ a partir de la ecuación (1.2), se busca minimizar:

$$\min_{\forall h \in R^L} ||S\vec{h} - \vec{r}|| \tag{2.1}$$

Por lo que se envía una secuencia conocida de $S$ de prueba para cuantificar el error sistemático impulsivo sobre la señal original. La norma de la ecuación 2.1 es norma dos. Se aclara que se realiza una aproximación, se busca el vector $\vec{h}$ que minimiza ese error — pero como el sistema está influenciado por un proceso estocástico nunca se va a hallar la respuesta impulsiva exacta (que depende del material y de la señal enviada). Teniendo esto último en cuenta, consideramos entonces la ecuación:

$$\vec{r} = S\vec{h} \tag{2.2}$$

Para utilizar el método de Cholesky en una ecuación del tipo $\vec{b} = A\vec{h}$ es necesario que la matriz $A$ sea cuadrada, simétrica y definida positiva. Pero en nuestro caso, en su lugar tenemos a la matriz $S$, que como se observa en la ecuación 1.2 es triangular inferior y no es cuadrada. Como necesitamos una matriz que cumpla estas condiciones, multiplicamos ambos lados de la ecuación (2.2) por $S^T$:

$$S^T\vec{r} = S^T S\vec{h} \tag{2.3}$$

Llamamos $S^T\vec{r} = \vec{b}$ y $S^T S = A$, y obtenemos la expresión:

$$\vec{b} = A\vec{h} \tag{2.4}$$

En esta nueva ecuación (2.4), la matriz A la conforma el producto que $S^T$ por $S$, por lo que ahora además de ser cuadrada, la norma usual definida por la matriz $SS^T$ es siempre mayor a cero dado que los bits de Lena resultan ser así. Por consiguiente, se cumplen las hipótesis de Cholesky y $A$ es definida positiva para toda señal que se escriba de la forma que previamente se definió en el marco teórico del sistema. Para implementar este método, el cual consiste en encontrar las matrices $L$ y $L^T$ triangulares, tales que $A = LL^T$,

para simplificar la resolución de la ecuación. Una vez averiguadas las matrices $L$ y $L^T$,en la ecuación (2.4) reemplazamos la matriz $A$ por el producto entre ellas y obtenemos:

$$\vec{b} = LL^T\vec{h} \tag{2.5}$$

Llamamos $\vec{y} = L^T\vec{h}$ y lo reemplazamos en la ecuación (2.5), de modo que:

$$\vec{b} = L\vec{y} \tag{2.6}$$

A partir de la ecuación (2.6) obtenemos $\vec{y}$ realizando sustitución Forward. Finalmente, dado que, como se mencionó previamente:

$$\vec{y} = L^T\vec{h} \tag{2.7}$$

se despeja $\vec{h}$ de la ecuación (2.7) haciendo una sustitución Backward. La ventaja de implementar el método de Cholesky para la ecuación resuelta es que se terminan resolviendo dos ecuaciones en las que la matriz involucrada (ya sea $L$ o $L^T$) es una matriz triangular.

## 2.2 Estimación de la imagen original

Una vez usada la secuencia de entrenamiento para tener un estimador para la respuesta impulsiva H, se procede a minimizar el error cuadrático medio del estimador de la imagen de Lena. La estimación de la imagen original $\vec{s}$, se hace a partir de la ecuación (1.1), de una manera distinta a la implementada para la estimación de $\vec{h}$. A pesar de que también se realiza haciendo cuadrados mínimos, asumimos que el nivel de ruido es conocido y usamos otro método conocido como Linear Minimum Square Error (LMMSE); donde las deducciones de las fórmulas que se usan para hallar el estimador vienen de plantear una suerte de regresión lineal; y luego buscar maximizar la probabilidad según Maximum Likelihood Estimation (si se modela con Estadística Clásica o Frecuentista) o Maximum a Posteriori Probability Estimation (si se modela con Estadística Bayesiana). Si bien no es lo correcto, asumimos que las mínimas unidades transmitidas en la señal enviada son independientes entre sí, simplemente para que la resolución sea más sencilla.
Para ver cómo funciona todo lo explicado hasta aquí, se envía la imagen de Lena con un ruido de un desvío estándar que se va variando para cada experimento,, y para dos valores de $E$ distintos. Siendo $E$ la longitud de la secuencia de entrenamiento. A continuación se observan los resultados de la transmisión de la imagen de Lena en escala de grises.

Imagenes con $\sigma$= 1 y E= 512

Original | Recibida | Estimada

Imagenes con $\sigma$= 0 y E= 32

Original | Recibida | Estimada

Imagenes con $\sigma$= 0.1 y E= 32

Original | Recibida | Estimada

Imagenes con $\sigma$= 10 y E= 32

Original

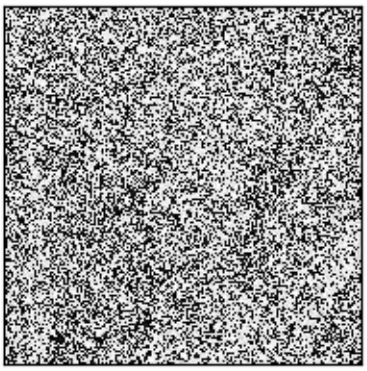
Recibida

Estimada

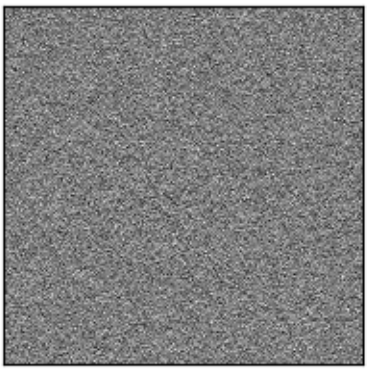
Imagenes con $\sigma$= 20 y E= 32

Original

Recibida



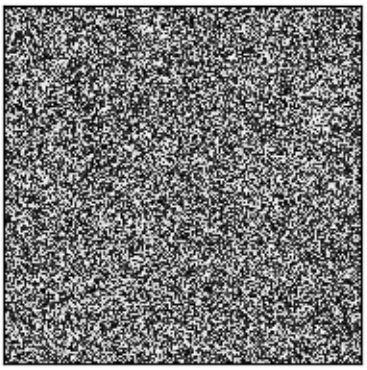
Estimada



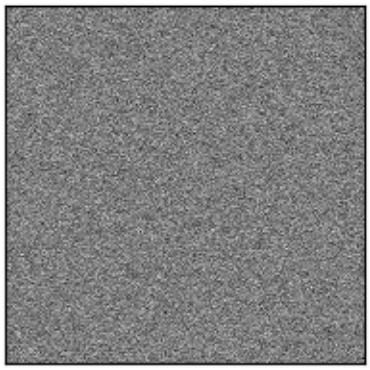
Imagenes con $\sigma$= 50 y E= 32

Original

Recibida



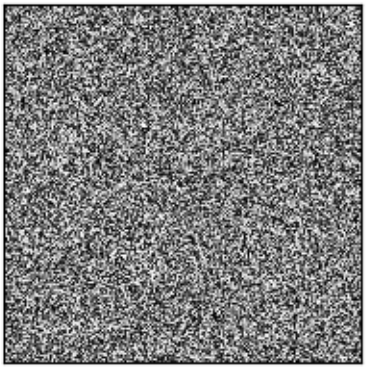
Estimada



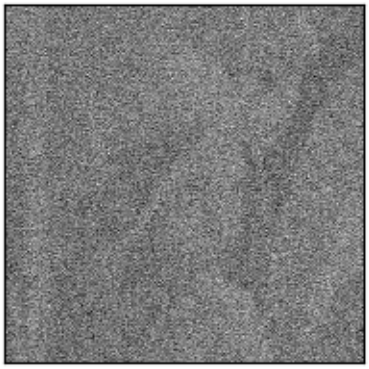
Imagenes con $\sigma$= 100 y E= 32

Original

Recibida



Estimada



Imagenes con $\sigma$= 0 y E= 1024

Original

Recibida

Estimada

Imagenes con $\sigma$= 0.1 y E= 1024

Original

Recibida

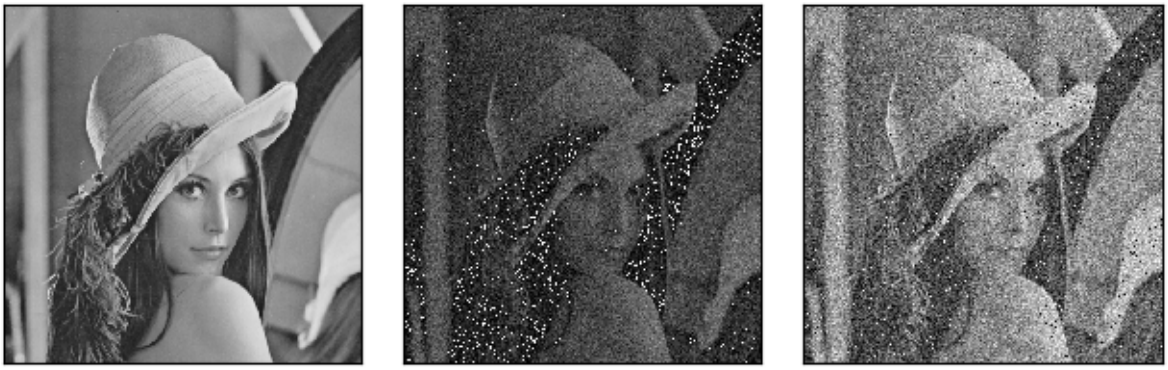
Estimada

Imagenes con $\sigma$= 10 y E= 1024

Original | Recibida | Estimada

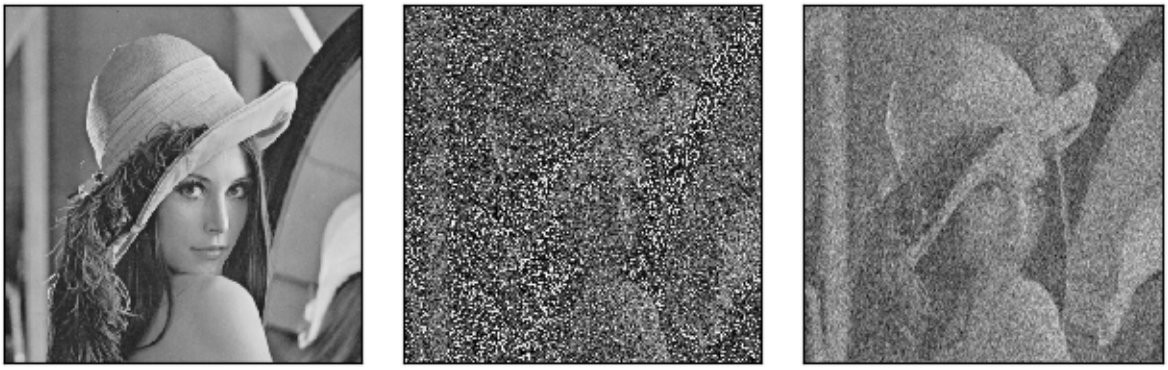
Imagenes con $\sigma$= 20 y E= 1024

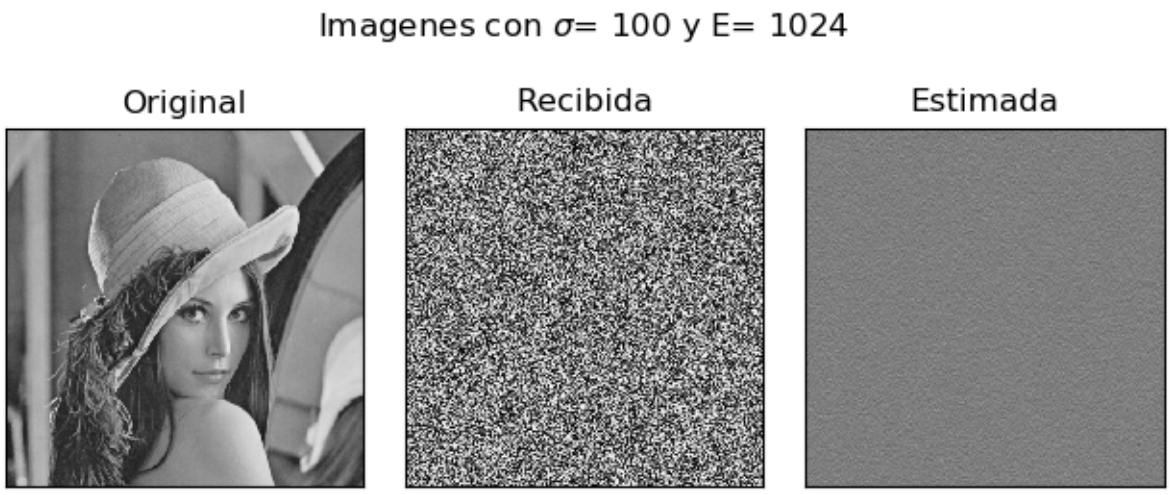
Original | Recibida | Estimada



Imagenes con $\sigma$= 50 y E= 1024

Original | Recibida | Estimada






## 3 Conclusión - Análisis de imágenes

Tras ascender la desviación estándar del ruido gaussiano, provoca que el estimador resultante sea muy distinto respecto de la señal original. Aunque los efectos impulsivos se puedan cuantificar con una señal de prueba, estos no están aislados del ruido Gaussiano, y por ende genera desviaciones agregadas además del $N_k$ en sí que son difíciles de solventar, aun asumiendo conocido el nivel de ruido. La ventaja entonces del uso de la señal de entrenamiento es aprovecharse del hecho de que, como la respuesta impulsiva $h_k$ depende del medio de comunicación (con su parámetro de longitud) y la señal que con cada experimento no cambia (entiéndase por experimento a cuando se genera un solo h particular, que vendría a estar asociado a cambiar el medio y la señal), quiere decir que $h_k$ es determinista y el aporte aleatorio viene dado por $N_k$. Esto explica el efecto de por qué tras aumentar la cantidad de información, $E$, no afecta demasiado en comparación a cambiar el desvío estándar del ruido gaussiano, por lo menos en el rango estudiado. Sí puede apreciarse de todas maneras, que acrecentar el $E$ conlleva a aumenta el error cuadrático medio y por ende la imagen original tiende a distinguirse aún más de la recuperada por el estimador.

## 4 Anexo

A continuación se presenta el código empleado para realizar las estimaciones y graficar las imagenes de este trabajo. Aclaración: en utils.py se encuentran las funciones que había que implementar. Se adjuntan también los .py del código que se muestran a continuación. En caso de ejecutarlo, es necesario correr ej1.py teniendo en el mismo directorio utils.py y lena.bmp.

```
from utils import *
from numpy import *
from numpy.random import *
from matplotlib.pyplot import *
from scipy.linalg import toeplitz
from numpy.linalg import inv

E = 32
sigma = 0.1
L = 5
ganancia = 1/10
h = ganancia*(1+randn(1,L))
```

```
a = imread(("lena512.bmp"))
M = len(a[:, 0])
P = len(a[0, :])
z = np.zeros(M-L)
H = toeplitz(np.concatenate((h,z),axis=None),zeros(M))
r = zeros((M,P))
N = sigma*randn(M,P)
s = a.astype(float64)
r = matmul(H,s) + N
b = uint8(r)
# #-------------------------------

sE = np.random.uniform(1,255,size=(E,1))-1
ME= len(sE[:, 0]) #cols
PE= len(sE[0, :]) # fils
#ajusto sE para el producto

HE= toeplitz(concatenate((h,zeros(ME-L)),axis=None),zeros(ME))
rE= zeros((ME,PE))
NE= sigma*randn(ME,PE)
rE = matmul(HE,sE) + NE
S = toeplitz(sE.transpose())

#S = S(:,1:L) # en octave dame todas las columnas de 1 a L
S= S[:,0:L+1] # en python 0:L+1 incluye hasta L

S = tril(S)
# multiplicamos ambos miembros por izquierda para obtener una matriz
# simetrica y aplicar cholesky
St = S.transpose()

he=solveEq(matmul(St,S),matmul(St,rE))

#comparo h con hE
compareh= [h,he]
print(compareh)
He= toeplitz(concatenate((he,zeros(M-L-1)),axis=None),zeros(M))

CN= (sigma**2)*eye(len(He))

sigmax= (1/256*(sum(np.power (((range(0,255))-mean(range(0,255))),2))))**(1/2)
mx= mean(range(0,255))*ones(M)
CX= (sigmax**2)*eye(len(He))

aux = matmul(matmul(He,CX),He.transpose())+CN
W = matmul(matmul(CX,He.transpose()),inv(aux))
#-------------------------------------------------------------

d = zeros((M,P))
for k in range(0,P):
    #d(:,k) accede en octave a la columna k-esima
    d[:, k] = W.dot((r[:, k])-He.dot(mx)) + mx

d = uint8(d)

```

```
68
69 #graficamos
70
71 figure(2)
72
73 suptitle("Imagenes con $\sigma$= "+str(sigma)+" y E= "+str(E))
74
75 subplot(1,3,1)
76 imshow(a,cmap='gray',vmin=0,vmax=255)
77 yticks([])
78 xticks([])
79 title("Original")
80
81 subplot(1,3,2)
82 yticks([])
83 xticks([])
84 imshow(b,cmap='gray',vmin=0,vmax=255)
85 title("Recibida")
86
87 subplot(1,3,3)
88 imshow(d,cmap='gray',vmin=0,vmax=255)
89 yticks([])
90 xticks([])
91 title("Estimada")
92 tight_layout()
93 name = "E" + str(E) + "S" + str(sigma)
94 show()
```

Listing 1: Ej1.py

---

```
1  import numpy as np
2
3  def cholesky(a):
4      #esta funcion asume que a es es semidefinida positiva y simetrica
5      #(esto esta justificado en el informe)
6      w = len(a)
7      l = np.zeros((w,w)) # matriz de wxw
8      #sum1 y sum2 son las sumatorias de las formulas
9      sum1 = 0
10     sum2 = 0
11     #j es columna, i es fila
12     #como lo primero que termina una iteracion en el proceso son las filas
13     #entonces deben estar adentro del for de las columnas
14     for j in range(0, w):
15         for i in range(j, w):
16             if i == j: #formula para cuando i vale j
17                 for k in range(0, i):
18                     sum1 += (l[i][k]) ** 2
19                 l[i][i] = (a[i][i] - sum1) ** (1 / 2)
20                 sum1 = 0
21             else:      #formula para cuando i distinto j
22                 for k in range(0, j):
23                     sum2 += l[i][k]*l[j][k]
```

```
                l[i][j] = (a[i][j]-sum2)/(l[j][j])
                sum2 = 0
    return l

def backSubs(u,y):
    # queremos resolver u x = y
    # con a triangular superior
    n = len(u)
    x = np.zeros(n)
    sum = 0
    for i in range(n-1,-1,-1): # [n-1,0]
        for j in range(i+1,n):
            sum += x[j]*u[i][j]
        x[i]=(y[i]-sum)/u[i][i]
        sum = 0
    return x

def ForwSubs(u,y):
    # queremos resolver u x = y
    # con u triangular inferior
    n=len(u)
    x =np.zeros(n)
    sum = 0
    for i in range(0,n): # n a 1
        for j in range(0,i):
            sum += x[j]*u[i][j]
        x[i]=(y[i]-sum)/u[i][i]
        sum = 0
    return x


def solveEq(a,b):
    #solve ax = b
    #hacemos cholesky
    L = cholesky(a)
    #primero resolvemos L y = b
    y = ForwSubs(L,b)
    #luego resolvemos (L)t x = y
    x = backSubs(L.transpose(),y)
    return x
```

Listing 2: utils.py