IEE239 - Procesamiento de Señales e Imágenes Digitales
Laboratorio 1 - Tarea Extra
Primer Semestre 2018

- Puntaje total: 6 puntos adicionales.
- Fecha límite de entrega: Viernes, 13 de abril a las 23:59.
- **La evaluación es estrictamente personal. Cualquier falta de probidad será sancionada con la nota desaprobatoria de cero en la sesión de laboratorio.**

Instrucciones:

1. Revisar el material de referencia **MIT OpenCourseware: Introduction to Matlab**[1]
2. A partir de las **lecturas 1**[2] **y 2**[3], desarrollar los siguientes ejercicios computacionales.
3. Presentar el código desarrollado y comentado en la carpeta `/laboratorio/lab01/'horario'/tarea/` bajo el formato `lab01_tarea_'codigo pucp'.zip`. El archivo debe contener el script principal, así como las 2 subrutinas de la pregunta 2:
   a. `lab01_tarea_'codigo pucp'.m`
   b. `func_encriptar.m`
   c. `func_desencriptar.m`
4. Separar cada pregunta usando **Code Sections**[4].

Puntaje:

Cada pregunta tiene un valor de *1 punto*, excepto la pregunta 3 cuyo valor es de *2 puntos*.

[1] http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-094-introduction-to-matlab-january-iap-2010/index.htm
[2] http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-094-introduction-to-matlab-january-iap-2010/lecture-notes/MIT6_094IAP10_lec01.pdf
[3] http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-094-introduction-to-matlab-january-iap-2010/lecture-notes/MIT6_094IAP10_lec02.pdf
[4] http://www.mathworks.com/help/matlab/matlab_prog/run-sections-of-programs.html

1. Operaciones con Escalares, Vectores y Matrices

1.1. Variables Escalares

Crear las siguientes variables:

i. $a = 10$.

ii. $b = \pi^{1.5}$.

iii. $c = e^{j(\pi + \frac{1}{4})}$, donde j es el número imaginario y e es el número de Euler. Usar `exp()`, `pi`.

iv. $d = \text{tg}^{-1}\{0.45\}$, donde las unidades del argumento corresponden a radianes. Usar `atan()`.

1.2. Variables Vectoriales

Crear las siguientes variables:

i. $\text{aVec} = \begin{bmatrix} 2j & 5 & e^{j\pi} & \frac{1}{3} \end{bmatrix}$.

ii. $\text{bVec} = \begin{bmatrix} 1 \times 10^3 \\ 0.1 \\ \cos(\frac{\pi}{2} - \frac{1}{4}) \\ 50 \end{bmatrix}$.

iii. $\text{cVec} = \begin{bmatrix} 10 & 9.75 & \cdots & -9.75 & -10 \end{bmatrix}$, la secuencia de números de 10 hasta -10 con pasos de 0.25.

iv. $\text{dVec} = \begin{bmatrix} 2^3 & 2^{2.5} & \cdots & 2^{-2.5} & 2^{-3} \end{bmatrix}$.

v. eVec = Jacobiano, eVec es del tipo `string` o arreglo de caracteres.

1.3. Variables Matriciales

Crear las siguientes matrices a partir de operaciones vectoriales y rutinas de MATLAB. **No crear matrices de forma explícita**:

i. Crear la variable aMat como una matriz de 15×15 cuya séptima fila corresponde al vector $\{-7, -6, \ldots, 6, 7\}$, séptima columna corresponde al vector $\{-14, -12, \ldots, 12, 14\}^T$ y el resto de elementos son 0. Usar `zeros()`.

ii. $\text{bMat} = \begin{bmatrix} 1 & 14 & 1 & \cdots & & 1 \\ 1 & 1 & 13 & \ddots & & \vdots \\ 1 & 2 & \ddots & \ddots & & \\ \vdots & \ddots & \ddots & & 2 & 1 \\ & & & 13 & 1 & 1 \\ 1 & & \cdots & 1 & 14 & 1 \end{bmatrix}$, matriz de 15×15 cuyas diagonales superior e inferior corresponden a los vectores $\{14, 13, \ldots, 1\}$ y $\{1, 2, \ldots, 14\}$, respectivamente, mientras que el resto de elementos son 1. Usar `diag()`.

iii. $\text{cMat} = \begin{bmatrix} x[0] & x[1] & \cdots & x[14] \\ x[15] & x[16] & \cdots & x[29] \\ \vdots & \vdots & \ddots & \vdots \\ x[210] & x[211] & \cdots & x[224] \end{bmatrix}$, matriz de 15×15 compuesta por los elementos del vector $x[n] = \cos\left(\frac{2\pi n}{225}\right)$. Usar `reshape()`:

iv. $\text{dMat} = \begin{bmatrix} 2 & 8 & 32 & 128 & 512 \\ 8 & 32 & 128 & 512 & 128 \\ 32 & 128 & 512 & 128 & 32 \\ 128 & 512 & 128 & 32 & 8 \\ 512 & 128 & 32 & 8 & 2 \end{bmatrix}$, matriz de 5×5 en la que cada columna se compone de elementos de la primera columna. Usar `hankel()`.

v. Crear la variable eMat como una martiz de 6×10 cuyos elementos son números enteros de 20 a 30 elegidos de forma aleatoria con distribución uniforme. Usar `rand()`.

vi. $\text{fMat} = \begin{bmatrix} 1 & 5 & \text{NaN} & 13 \\ 2 & \text{NaN} & 10 & \text{Inf} \\ \text{NaN} & \text{Inf} & 11 & \text{NaN} \\ 4 & 8 & \text{NaN} & 16 \end{bmatrix}$,

matriz de 4×4 con elementos del 1 a 16, donde los múltiplos de 3 son reemplazados por el valor `Not a Number (NaN)` y los múltiplos de 7 son reemplazados por el valor Infinito `(Inf)`. Usar `mod()`.

1.4. Ecuaciones con Escalares

Crear las siguientes variables en función a los valores calculados en 1.1:

i. $x = \log_{\frac{1}{3}}\{\frac{a}{b}\}$. Usar `log10()` y propiedades de la función logaritmo.

ii. $y = \frac{\lceil b \rceil !}{\sqrt{-\mathbf{I}\{c\}}}$, donde $\lceil x \rceil$ corresponde a la función techo (*roof*) de x, $x!$ indica el factorial de x e $\mathbf{I}\{x\}$ indica la componente imaginaria de x. Usar `ceil()`, `factorial()`, `imag()` y `sqrt()`.

iii. $z = \frac{(e^c)^* \cdot ja}{\sqrt{b+jc}}$, donde j corresponde al número imaginario y x^* corresponde al complejo conjugado de x. Usar `conj()`.

1.5. Ecuaciones con Vectores

A partir de las variables creadas en 1.1 y 1.2, resolver las siguientes operaciones entre vectores. Tener presente que los operadores .*, ./ y .^ producen operaciones elemento a elemento.

a. $\text{xVec} = \frac{1}{\sqrt{2\pi}} e^{\frac{(\text{aVec}-a)^2}{\text{bVec}^T}}$, donde bVec^T corresponde al vector bVec transpuesto. Usar `transpose()`.

b. $\text{yVec} = \frac{\text{aVec} \cdot \text{bVec}}{\sqrt{\text{aVec}^T + \text{bVec}}}$, donde $a \cdot b$ corresponde al producto vectorial entre a y b.

c. $\text{zVec} = \sin(\text{dVec} + 2)$.

1.6. Ecuaciones con Matrices

A partir de las matrices y vectores creados en 1.2 y 1.3, determinar las siguientes matrices:

a. $\text{xMat} = \text{aVec}^T \cdot \text{aVec}$.

b. $\text{yMat} = \text{aMat} \cdot \text{cMat} + \text{aMat} \odot \text{cMat}$, donde $A \odot B$ corresponde al producto elemento a elemento entre las matrices A y B.

c. $\text{zMat} = \exp\{\tan\{\text{dMat}\} \cdot \mathbf{1}_{5\times 15} \cdot \text{cMat}^T\}$, donde $\mathbf{1}_{M\times N}$ corresponde a una matriz de 1 de dimensiones $M \times N$.

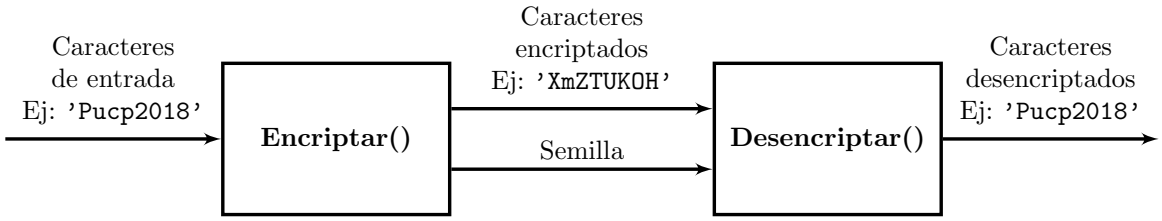


Figura 1: Sistema de encriptación de caracteres.

2. Uso de Subrutinas

Se propone implementar una modificación del método de encriptación de caracteres **Monoalphabetic Shifting Substitution Cyphers**[5]. El algoritmo contempla dos etapas:

- **Encriptar()**: La etapa de encriptación recibe como argumento de entrada la secuencia de interés, compuesta por caractéres alfanuméricos y de longitud arbitraria. **Solo se permite la encriptación de letras (mayúsculas o minúsculas) y números enteros positivos incluyendo** 0. A partir de la secuencia, se reemplaza cada caracter basándose en un ordenamiento pseudo-aleatorio de todos los posibles caracteres y se genera una secuencia encriptada. Adicionalmente, se preserva información del ordenamiento pseudo-aleatorio. Este último dato es denominado semilla o **random seed**[6] y brinda información esencial para desencriptar el mensaje.
- **Desencriptar()**: La etapa de desencriptación recibe como argumentos de entrada la secuencia encriptada y la semilla. A partir de ellas, determina el reordenamiento que con el que se generó la secuencia encriptada y revierte el efecto. El resultado corresponde a la secuencia inicial de caracteres alfanuméricos.

La Figura 1 describe el sistema de encriptación propuesto. A partir de ello, implementar ambas etapas del método y evaluarlas para múltiples secuencias de entrada.

a. Como secuencia de prueba, crear el arreglo de caracteres `ABCDwxyz0123` y almacenarlo en la variable `input_str`. Notar que el arreglo contiene los 3 tipos de caracteres permitidos. Luego, verificar su longitud y tipo de variable a partir del comando `whos` e incluir dicha información en sus comentarios:

```
1          input_str= 'ABCDwxyz0123';
```

b. Crear una función[7] denominada `func_encriptar()` que reciba como argumento de entrada a la secuencia de interés y que correpsonda a la primera etapa del método de encriptación. Tener en consideración el siguiente procedimiento:

   i. Convertir la secuencia de entrada a un arreglo de variables del tipo **double** a partir de `double()` y almacenarla en la variable `input_double`. La secuencia resultante contendrá los códigos ASCII[8] de cada elemento y debe ser de las mismas dimensiones que la secuencia original. Verificar que los códigos resultantes corresponden a los caracteres iniciales.

   ii. Dado que solo se permite la encriptación de letras (mayúsculas o minúsculas) y números enteros positivos incluyendo 0, generar un aviso de error en caso la secuencia de entrada contenga otro tipo de caracteres. Para ello, generar comparaciones lógicas entre `input_double` y los rangos ASCII de los 3 tipos de caracteres permitidos, de tal manera que se obtengan los vectores lógicos

[5] https://pdfs.semanticscholar.org/4ce0/f3f97dfa44d12c5805897994334b23490655.pdf.
[6] Los detalles pueden ser consultados en https://www.mathworks.com/help/matlab/ref/rng.html.
[7] Revisar la documentación relevante en https://www.mathworks.com/help/matlab/ref/function.html.
[8] El código de cada caracter puede ser consultado en http://www.columbia.edu/kermit/ascii.html.

`mask_upper`, `mask_lower` y `mask_num` para identificar mayúsculas, minúsculas y números enteros positivos incluyendo 0, respectivamente.

Por ejemplo, a partir del rango ASCII para letras, la operación lógica para determinar si los elementos son minúsculas puede ser realizada de la siguiente forma:

```
mask_lower= input_double≥ 97 & input_double≤ 122;
```

Sumar los 3 vectores lógicos y determinar si algunos de los caracteres no corresponde a ninguno de los 3 casos verificando si algun elemento en la secuencia resultante es 0. En caso sea así, generar el siguiente mensaje de error a partir de `error()`:

```
error( 'Error: Argumento de entrada contiene caracteres que no son ...
    letras o numeros enteros positivos incluyendo 0.');
```

iii. Crear el vector `ref_v` que contenga todos los posibles códigos ASCII de los elementos de entrada. Verificar los rangos de los 3 posibles tipos de caracteres en la tabla ASCII y que el vector resultante es de 62 elementos de longitud.

iv. Crear un orden pseudo-aleatorio para los elementos de `ref_v` y almacenar la semilla que generó dicho orden. Para ello, usar `randperm()` y el comando `rng`:

```
s = rng;     % semilla
randpos_v= randperm( ref_size); % ordenamiento pseudo-aleatorio
```

donde `ref_size` corresponde a la longitud de `ref_v`. Luego, usar los nuevos índices para crear un reordenamiento en la secuencia de posibles códigos ASCII y almacenarlo en la variable `randref_v`:

```
randref_v= ref_v( randpos_v); % posibles codigos ASCII reordenados
```

v. Inicializar el argumento de salida `coded_str` como un vector de 0 con las mismas dimensiones que la secuencia de entrada a partir de `zeros()`. Luego, almacenar en él los caracteres encriptados en función a `randref_v` a partir de un `for-loop`.

Por ejemplo, para encriptar el elemento i-ésimo de `input_str`, determinar la posición de su código ASCII en el vector `ref_v`. Luego, determinar qué código ASCII aparece en dicha posición en el vector `randref_v`. Finalmente, almacenar este nuevo valor ASCII en la posición i-esima del vector de salida `coded_str`.

vi. Convertir la nueva secuencia de códigos ASCII `coded_str` a un arreglo de caracteres a partir de `char()`. Esta nueva secuencia corresponde al mensaje encriptado.

La rutina debe proporcionar como argumentos de salida la secuencia encriptada `coded_str` y la semilla utilizada para crear el reordenamiento `s`.

c. Crear una función denominada `func_desencriptar()` que reciba como argumentos de entrada la secuencia encriptada y la semilla utilizada en su creación. Para ello, tener en consideración el siguiente procedimiento:

i. Convertir la secuencia `coded_str` en sus códigos ASCII a partir de `double()`. Verificar que los códigos corresponden a los caracteres.

ii. Similar a la etapa anterior, crear un vector `ref_v` que contenga todos los posibles códigos ASCII de los elementos de entrada. Tener en consideración que `ref_v` debe tener el mismo orden que la versión usada en `encriptar()`. Verificar que el vector resultante tiene una longitud de 62 elementos.

iii. A partir de la semilla **s** ingresada como segundo argumento, generar el reordenamiento utilizado en la encriptación **randpos_v** a partir de **randperm()** y el comando **rng**:

```
rng(s)
randpos_v= randperm( ref_size); % reordenamiento usado en la etapa de ...
    encriptacion
```

Donde **ref_size** corresponde a la longitud de **ref_v**. Luego, usar los nuevos indices para crear nuevamente el vector de códigos ASCII reordenado **randref_v**:

```
randref_v= ref_v( randpos_v); % secuencia ASCII con mismo ...
    reordenamiento que la etapa de encriptacion.
```

iv. Inicializar el argumento de salida **decoded_str** como un vector de 0 con las mismas dimensiones que **coded_v** a partir de **zeros()**. Luego, almacenar en él los caracteres desencriptados en función a **ref_v** a partir de un **for-loop**.

Por ejemplo, para el elemento i-ésimo de la secuencia **coded_str**, determinar la posición de su código ASCII en el vector **randref_v**. Luego, determinar qué código ASCII aparece en dicha posición en el vector **ref_v**. Finalmente, almacenar este nuevo valor ASCII en la posición i-esima del vector de salida **decoded_str**.

v. Convertir la nueva secuencia de códigos ASCII **decoded_str** a un arreglo de caracteres a partir de **char()**. Esta nueva secuencia corresponde al mensaje desencriptado.

d. Demostrar el funcionamiento del sistema ante la secuencia de prueba. Luego, evaluar su funcionamiento para las siguientes secuencias:

- 'Pseudoinversa'
- 'e=mc2trooper'
- 'Pucp2018'

Para cada caso, mostrar en línea de comandos el resultado:

```
Secuencia de entrada: Pseudoinversa
Secuencia encriptada: DkbpmXv2JbfkM
Secuencia desencriptada: Pseudoinversa
```

```
Secuencia de entrada: e=mc2trooper
Error using func_code
Error: Argumento de entrada contiene caracteres que no son letras o ...
    numeros enteros positivos.
```

```
Secuencia de entrada: Pucp2018
Secuencia encriptada: XmZTUKOH
Secuencia desencriptada: Pucp2018
```

3. Funciones Comunes e Indexado

La imagen a colores **i03.jpg**[9] está representada por tres capas de color: la primera capa contiene las intensidades de color rojo (capa **R**), la segunda contiene las intensidades de color verde (capa **G**) y la

[9]La imagen está includa en la carpeta /laboratorio/lab01/tarea_extra

(a) (b) (c) (d)

Figura 2: Imagen a colores en espacio RBG. (a) Imagen a colores. (b) Capa roja. (c) Capa verde. (d) Capa azul.

tercera contiene las intensidades de color azul (capa **B**). La imagen está almacenada como una variable del tipo `uint8` (entero sin signo de 8 bits), por lo que cada capa solo puede tomar valores enteros entre 0 y 255. A partir de esto, se propone realizar un conjunto de operaciones aritémticas sobre sus capas.

a. Leer la imagen y almacenarla en la variable `i01_mat` a partir de `imread()`. Luego, usar el comando `whos` y verificar que se trata de una matriz de 886 filas, 602 columnas y 3 capas del tipo `uint8`.

b. En una misma ventana, Graficar la imagen de interés y las 3 capas que la componen a partir de `imshow()`, `plot()` y `subplot()`. La Figura 2 muestra el resultado esperado.

c. Como introducción a la extracción de bits, generar un número aleatorio entero con distribucion uniforme de 0 a 15 a partir de `rand()`. Luego, obtener el valor del bit 4 (donde el bit 0 es el menos significativo y el bit 7 es el más significativo) a partir de `bitget()`. Repetir el procedimiento para extraer el bit 7 y verificar que los bits obtenidos son coherentes con el valor aleatorio.

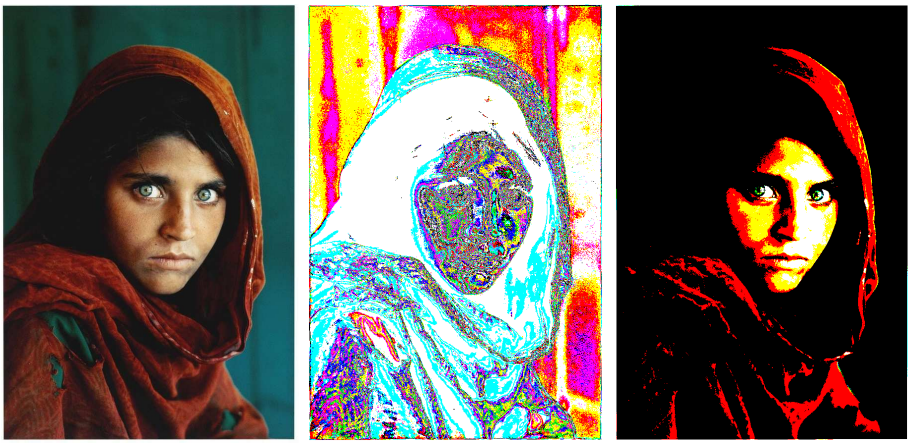
d. Similar al inciso anterior, se propone verificar la extracción de bits en matrices. Obtener el bit 4 de las 3 capas de `i01_mat` a partir de `bitget()`. El resultado debe ser una matriz binaria de dimensiones $886 \times 602 \times 3$. En una nueva ventana, graficar el resultado a partir de `imshow()`[10]. Luego, repetir el proceso para el bit 7. La Figura 3 muestra los resultados esperados. De ambos bits, cuál contiene más información acerca de la **estructura** de la imagen? Justificar claramente su respuesta e incluirla en sus comentarios.

e. El archivo **i03.mat**[11] contiene 8 matrices binarias que representan a los 8 bits de las 3 capas de `i01_mat`, denominadas $\{X_k\}_{k=0}^{7}$. Se sabe que las matrices $\{X_0, \ldots, X_3\}$ corresponden a los bits $\{0, \ldots, 3\}$, respectivamente. En cambio, las matrices $\{X_4, \ldots, X_7\}$ representan a los 4 bits más significativos pero han sido ordenadas aleatoriamente. Para determinar a qué bit corresponde cada matriz, se propone el siguiente método:

   i. Leer **i03_vars.mat** a partir del comando `load` de la siguiente forma:[12]

```
1            load i03_vars.mat
```

[10] Multiplicar la matriz por 255 antes de usar `imshow()` para obtener una escala adecuada.
[11] El archivo está incluido en la carpeta /laboratorio/lab01/tarea_extra.
[12] Asumiendo que el archivo está almacenado en el directorio actual en el entorno Matlab.



(a) (b) (c)

Figura 3: Plano de bits. (a) Imagen a colores en espacio RGB. (b) Bit 4 de las 3 capas RGB. (c) Bit 7 de las 3 capas RGB.

Crear una matriz `I_lsb` compuesta por los 4 bits menos significativos de `i01_mat`. Para ello, considerar las matrices $\{X_0, \ldots, X_3\}$, multiplicarlas por el peso de su bit correspondiente y sumarlas:

$$\texttt{I\_lsb} = w_0 \cdot X_0 + ... + w_3 \cdot X_3,$$

donde $w_k = 2^k$ corresponde al peso de cada bit.

ii. Crear un arreglo de celdas (**Cell Array**)[13] denominado `X_cell` de dimensiones 4×1 y almacenar las matrices de los 4 bits mas significativos. El arreglo debe estar ordenado de la siguiente forma:

```
X_cell{ 1}= X4;
X_cell{ 2}= X5;
X_cell{ 3}= X6;
X_cell{ 4}= X7;
```

iii. Para determinar el orden correcto de $\{X_4, \ldots, X_7\}$, primero se obtendrán todas sus posibles combinaciones. A partir de sus posiciones en `X_cell`, generar la matriz de permutaciones `C_mat` usando la rutina `perms()`. Verificar que la matriz resultante es de dimensiones 24×4 y contiene todos los posibles órdenes:

```
idx= 1: 4;    % indices a ordenar
C_mat= perms( idx);     % posibles permutaciones
```

iv. Desarrollar una búsqueda exhaustiva para determinar el orden correcto de $\{X_4, \ldots, X_7\}$ a partir del pseudocódigo descrito en 1[14], donde vec(X)[15] corresponde a la versión vectorizada de la matriz X y $d_2\{\mathbf{x}, \mathbf{y}\}$[16] corresponde a la distancia Euclidiana entre los vectores $\mathbf{x}$ e $\mathbf{y}$, ambos de longitud N:

$$d_2\{\mathbf{x}, \mathbf{y}\} \triangleq \sqrt{\sum_{i=0}^{N-1} (x_i - y_i)^2}.$$

[13] https://www.mathworks.com/help/matlab/matlab_prog/what-is-a-cell-array.html
[14] El pseudocódigo asume a 0 como indice inicial, mientras que los indices en Matlab inician en 1.
[15] revisar el operador `(:)`.
[16] revisar la rutina `norm()`.

Algoritmo 1 Determinar el orden correcto de las matrices $\{X_4, \ldots, X_7\}$.

Entrada: $P = 24$: número de permutaciones; I^*: imagen original; I_{LSB}: imagen compuesta por los 4 bits menos significativos; C: matriz de combinaciones; X_{cell}: arreglo con las 4 matrices de interés.

Salida: I_{rec}: Imagen reconstruida con el orden correcto de matrices.

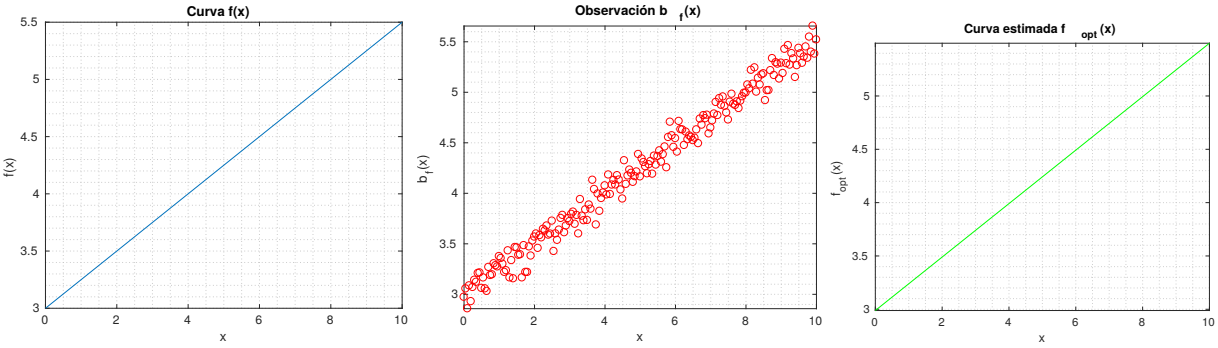
for $i = 0$ to $P - 1$ **do**
  $\hat{I} \leftarrow I_{\text{LSB}}$ ▷ Inicializar acumulador
  for $j = 0$ to 3 **do** ▷ Índices de las 4 matrices en X_{cell} (asumiendo indice inicial 0)
    $w \leftarrow 2^{j+4}$ ▷ Peso del bit actual
    $a \leftarrow C(i, j)$ ▷ Indice actual de la combinación
    $\hat{I} \leftarrow \hat{I} + w \cdot X_{\text{cell}}\{a\}$ ▷ Acumular
  end for
  $\varepsilon(i) \leftarrow d_2\{\text{vec}(\hat{I}), \text{vec}(I^*)\}$ ▷ Evaluar similaridad
end for

$i_{\text{true}} \leftarrow \underset{i}{\text{argmin}} \quad \varepsilon(i)$ ▷ Orden correcto: aquel que minimiza la métrica. Revisar `min()`.
$B_4 \leftarrow X_{\text{cell}}\{C(i_{\text{true}}, 0)\}$ ▷ Bit 4
$B_5 \leftarrow X_{\text{cell}}\{C(i_{\text{true}}, 1)\}$ ▷ Bit 5
$B_6 \leftarrow X_{\text{cell}}\{C(i_{\text{true}}, 2)\}$ ▷ Bit 6
$B_7 \leftarrow X_{\text{cell}}\{C(i_{\text{true}}, 3)\}$ ▷ Bit 7
$I_{\text{rec}} \leftarrow I_{\text{LSB}} + 2^4 \cdot B_4 + \cdots + 2^7 \cdot B_7$ ▷ Imagen reconstruida

(a) (b)

Figura 4: Imagen reconstruida. (a) Orden incorrecto de 4 bits más significativos. (b) Orden correcto de 4 bits más significativos.

La distancia Euclidiana es una medida de similitud entre dos vectores. A mayor similitud entre ellos, menor es su distancia. Una vez determinado el orden de las matrices, graficar la imagen resultante a partir de `imshow()` y verificar que es idéntica a la original. La Figura 4 muestra un contraste entre una reconstrucción con el orden incorrecto de bits y aquella con orden correcto.



(a) (b) (c)

Figura 5: Regresión lineal de una parábola. (a) Búsqueda de línea para $c = 10^{-4}$. (b) Búsqueda de línea para $c = 1.5 \times 10^{-3}$. (c) Curva ideal, observación y curva estimada.

4. Regresión Lineal

Se cuenta con las funciones unidimensionales $\{f(x), g(x)\}$, las cuales corresponden a una recta y una parábola, respectivamente. Se cuenta además con sus versiones distorsionadas u **observaciones** $\{b_f(x), b_g(x)\}$:

$$f(x) = 0.25x + 3, \tag{1}$$
$$g(x) = -4x^2 + 12x + 1, \tag{2}$$

$$b_f(x) = f(x) + \eta(x),$$
$$b_g(x) = g(x) + \eta(x),$$

donde η corresponde a una señal aleatoria con distribución Gaussiana de media 0 y desviación estandar 0.1: $\eta \sim \mathcal{N}(\mu = 0, \sigma = 0.1)$. A partir de las observaciones, se propone estimar los argumentos de cada función basándose en un **problema de optimización**.

a. Crear el vector $x_{\text{ideal}} \in [0, 10]$ con pasos de $\frac{1}{100}$ y la función $f(x_{\text{ideal}})$ de acuerdo a (1). Graficar el resultado y rotularlo adecuadamente a partir de `plot()`, `xlabel()`, `ylabel()`, `title()`, `axis` y `grid`. La Figura 5a muestra el resultado esperado.

b. Crear el vector $x_{\text{obs}} \in [0, 10]$ con pasos de $\frac{1}{20}$ y la observación $b_f(x_{\text{obs}})$ a partir de lo siguiente:

   i. Crear el vector $f(x_{\text{obs}})$.

   ii. Crear el vector η de las mismas dimensiones de $f(x_{\text{obs}})$ a partir de `randn()`[17].

   iii. Crear $b_f(x_{\text{obs}})$ a partir de la suma de ambos vectores.

   Graficar el resultado usando marcadores circulares de color rojo[18] y rotularlo adecuadamente. La imagen 5b muestra el resultado esperado.

c. Para estimar los argumentos de $f(x)$, se propone minimizar la siguiente **función costo**:

$$\mathcal{F}(\alpha, \beta) = \frac{1}{2} \sum_{i=0}^{N-1} \left(\hat{f}(\alpha, \beta, x_i) - b_f(x_i)\right)^2,$$

[17] Revisar la documentación `randn()` para modificar la media y desviación estandar de la secuencia aleatoria.
[18] Revisar la documentación de `plot()` para modificar los parámetros de los marcadores.

donde el modelo $\hat{f}$, los argumentos de interés $\{\alpha, \beta\}$ corresponden a $\hat{f}(\alpha, \beta, x) = \alpha x + \beta$ y x_i corresponde a elementos del vector x_{obs} de longitud N. Visto como una operación martiz-vector, la función costo es expresable como:

$$\mathcal{F}(\mathbf{a}) = \frac{1}{2}\big(\mathbf{X}\mathbf{a} - \mathbf{b}\big)^T\big(\mathbf{X}\mathbf{a} - \mathbf{b}\big),$$

donde $\mathbf{A}^T$ corresponde a la transposición de la matriz $\mathbf{A}$ y:

$$\mathbf{X} = \begin{pmatrix} x_0 & 1 \\ \vdots & \vdots \\ x_{N-1} & 1 \end{pmatrix}_{N\times 2}, \quad \mathbf{a} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}_{2\times 1} \quad \mathbf{b} = \begin{pmatrix} b(x_0) \\ \vdots \\ b(x_{N-1}) \end{pmatrix}_{N\times 1}$$

Ya que $\mathcal{F}(\mathbf{a})$ es una función cuadrática y tiene un solo punto estacionario (mínimo global), los argumentos originales $\hat{\mathbf{a}} = (\hat{\alpha}, \hat{\beta})^T$ pueden ser estimados igualando su primera derivada a 0. Esto equivale a resolver el siguiente sistema de ecuaciones:

$$\begin{aligned} \nabla\mathcal{F}(\mathbf{a}) &= \mathbf{X}^T\mathbf{X}\mathbf{a} - \mathbf{X}^T\mathbf{b} = 0 \\ \hat{\mathbf{a}} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{b}. \end{aligned} \tag{3}$$

A partir de (3), calcular $\{\hat{\alpha}, \hat{\beta}\}$. Primero, crear la matriz $\mathbf{X}$ a partir del vector x_{obs}. Luego, resolver el sistema de ecuaciones de la siguiente forma[19]:

```
a_hat=( transpose( X)* X)\ transpose( X)* b_v( :);
alpha_hat= a_hat( 1);
beta_hat= a_hat( 2);
```

Graficar el resultado obtenido y rotularlo adecuadamente. Verificar que los valores obtenidos $\{\hat{\alpha}, \hat{\beta}\}$ son muy ceranos a los originales. La Figura 5c muestra el resultado esperado.

d. Crear un nuevo vector $x_{\text{ideal}} \in [0, 3]$ con pasos de $\frac{1}{100}$ y crear la señal $g(x_{\text{ideal}})$.

e. Crear un nuevo vector $x_{\text{obs}} \in [0, 3]$ con pasos de $\frac{1}{20}$ y crear la observación $b_g(x_{\text{obs}})$ a partir de lo siguiente:

   i. Crear el vector $g(x_{\text{obs}})$ de acuerdo a (2).

   ii. Crear el vector η de las mismas dimensiones de $g(x_{\text{obs}})$ a partir de `randn()`.

   iii. Crear $b_g(x_{\text{obs}})$ a partir de la suma de ambos vectores.

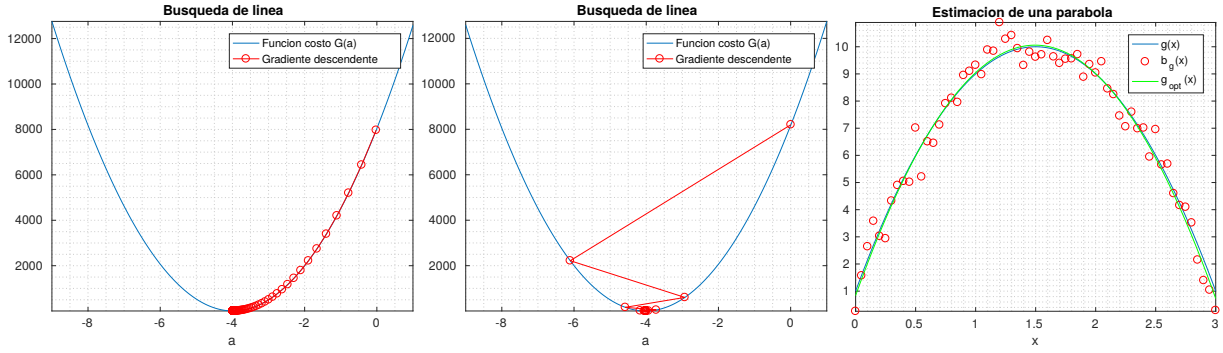
f. Solucionar el sistema lineal propuesto en (3) implica invertir una matriz, lo cual requiere de un gran número de operaciones matemáticas. Por ello, se propone usar un método más eficiente para estimar uno de los argumentos de $g(x)$ a partir de la función costo usada anteriormente:

$$\mathcal{G}(a) = \frac{1}{2}\sum_{i=0}^{N-1}\big(\hat{g}(a, x_i) - b_g(x_i)\big)^2,$$

donde el modelo $\hat{g}$ y el argumento a corresponden a $\hat{g}(a, x) = ax^2 + 12x + 1$.

Verificar que $\mathcal{G}(a)$ corresponde a una función costo con un solo punto estacionario (mínimo global). Para ello, crear un vector $a_{\text{search}} \in [-9, 1]$ con pasos de $\frac{1}{100}$ y evaluar $\mathcal{G}(a_{\text{search}})$ a partir de un `for-loop` de la siguiente forma:

[19] El operador \ permite implementar de forma eficiente la inversa de una matriz. La documentación puede ser consultada en https://www.mathworks.com/help/matlab/ref/mldivide.html.



(a) (b) (c)

Figura 6: Regresión lineal de una parábola. (a) Búsqueda de línea para $c = 10^{-4}$. (b) Búsqueda de línea para $c = 1.5 \times 10^{-3}$. (c) Curva ideal, observación y curva estimada.

i. Implementar la función `g_handle` a partir de `function handles`[20] que reciba como entrada un argumento a y de como respuesta el vector $\{\hat{g}(a, x_0), \ldots, \hat{g}(a, x_{N-1})\}^T$:

```
g_handle=( a) a* x_obs.^( 2)+ 12* x_obs+ 1;
```

ii. Para el elemento i-esimo del vector `a_search`, calcular $\mathcal{G}(a)$:

```
G_v( i)= 0.5* sum( ( g_handle( a_search( i))- b_g).^2);
```

Graficar el vector $\mathcal{G}(a_{\text{search}})$ resultante y rotularlo adecuadamente. Usando la herramienta `zoom in`, verificar que el valor mínimo de la función costo se obtiene para un valor a muy cercano al original.

g. El método iterativo a utilizar es denominado **búsqueda de línea** (line search) y actualiza el valor a en el sentido opuesto de la gradiente, de tal manera que cada nuevo valor reduzca la función costo hasta alcanzar el valor mínimo. Implementar el método para itermax= 500, tol= 10^{-8} y $c = 10^{-4}$. Verificar que el valor estimado $\hat{a}$ es muy cercano al valor original. Luego, graficar las posiciones evaluadas por el método y superponerlas en la gráfica de la función costo del inciso f a partir de `plot()`, `legend()` y el comando `hold on`. El Pseudocódigo 2 describe el método propuesto y la Figura 6a muestra el resultado esperado. Dado que es de interés calcular el escalar a óptimo, la gradiente de la función costo corresponde a:

$$\nabla\mathcal{G}(a) = \frac{d}{da}\mathcal{G}(a) = \sum_{i=0}^{N-1} \big(\hat{g}(a, x_i) - b_i\big)x_i^2.$$

h. Repetir el procedimiento del inciso g para $c = 1.5 \times 10^{-3}$. La Figura 6b muestra el resultado esperado.

i. En una sola ventana, graficar la parábola original, la observación y la parábola estimada. Verificar que la curva estimada es muy cercana a la original. La Figura 6c muestra el resultado esperado.

5. Manipulando variables

El archivo **q05.csv**[21] contiene las calificaciones de 71 estudiantes inscritos en 2 horarios de un curso, las cuales contemplan 5 notas de laboratorio, 2 notas de presentaciones técnicas y 2 exámenes escritos.

[20] https://www.mathworks.com/help/matlab/matlab_prog/creating-a-function-handle.html
[21] Los datos están almacenados en formato **CSV** (Comma Separated Values).

Algoritmo 2 Búsqueda de línea.

Entrada: c: tamaño de paso, maxiters: número máximo de iteraciones, $\mathbf{b} = \{b_0, \ldots, b_{N-1}\}$: observaciones, $\mathbf{x} = \{x_0, \ldots, x_{N-1}\}^T$: dominio de las observaciones, tol: tolerancia para condición de parada.
Salida: $\hat{a}$: parámetro estimado, $\mathcal{G}$: función costo en los valores a analizados.

$\mathbf{a} \leftarrow \mathbf{0}_{\text{maxiters}\times 1}$ ▷ Inicializar vector de valores a evaluados
$a_0 \leftarrow$ valor aleatorio de dist. gaussiana ($\sim \mathcal{N}(0,1)$) ▷ Valor inicial aleatorio
$\mathcal{G} \leftarrow \mathbf{0}_{\text{maxiters}\times 1}$ ▷ Inicializar vector de función costo
$r \leftarrow 0$ ▷ Contador

repeat
  $r \leftarrow r + 1$
  $\mathcal{G}_{r-1} \leftarrow \frac{1}{2}\sum_{i=0}^{N-1}\left(\hat{g}(a_{r-1}, x_i) - b_i\right)^2$
  $\texttt{grad} \leftarrow \sum_{i=0}^{N-1}\left(\hat{g}(a_{r-1}, x_i) - b_i\right)x_i^2$ ▷ Gradiente en la posición a actual
  $a_r \leftarrow a_{r-1} - c \cdot \texttt{grad}$ ▷ Actualizar a
until $|a_r - a_{r-1}| <$ tol or $r \geq$ maxiters

$\hat{a} \leftarrow a_r$ ▷ Parámetro óptimo a estimado

Se propone calcular las estadísticas del semestre y obtener las calificaciones finales de ambos horarios.

a. Cargar el archivo y almacenar su contenido en la variable `raw_cell`. Para ello, importar el archivo a partir de `readtable()` y luego convertir la variable en un arreglo de celdas a partir de `table2array()`. el resultado debe ser un arreglo de celdas de dimensiones 71×11.

b. La columna 11 de `raw_cell` contiene la información de los alumnos retirados del curso. Antes de obtener estadísticas, eliminar de la lista a los retirados. A partir de `strcmp()` y operaciones lógicas, crear una máscara binaria de dimensiones 71×1 en la que '0' corresponda a los alumnos con la palabra **Retirado** en la columna 11 y '1' al resto. Luego, usar dicha máscara como índice para descartar a los alumnos retirados del curso y almacenar la información filtrada en la variable `valid_cell`.

c. La columna 1 del `valid_cell` contiene el horario al que pertenece cada alumno. Nuevamente a partir de `strcmp()`, organizar la información por horarios, de tal forma que las notas del horario **h01** estén almacenadas en el arreglo de celdas `h01_cell` y las notas del horario **h02** estén almacenadas en el arreglo de celdas `h02_cell`. Validar los arreglos resultantes si se sabe que el horario **h01** cuenta con 32 alumnos no retirados y el horario **h02** cuenta con 30 alumnos no retirados.

d. Calcular el promedio final de los alumnos de cada horario, si se sabe que la calificación se obtiene de la siguiente forma:

$$\text{Final} = \frac{30 \cdot \text{Laboratorio} + 20 \cdot \text{Presentación} + 25 \cdot \text{Examen 01} + 25 \cdot \text{Examen 02}}{100} \tag{4}$$

Donde:

i. Las 5 sesiones de laboratorio corresponden a las columnas $\{2, \ldots, 6\}$ del arreglo de celdas de cada horario. La nota *Laboratorio* se obtiene del promedio de las 4 mejores calificaciones.

ii. Las 2 presentaciones técnicas corresponden a las columnas $\{7, 8\}$ del arreglo de celdas de cada horario. La nota *Presentación* corresponde al promedio de ambas.

iii. El las notas *Examen 1* y *Examen 2* corresponden a las columnas $\{9, 10\}$ del arreglo de celdas de cada horario, respectivamente.

Para calcular el promedio de forma adecuada, crear variables `calif01_mat` y `calif02_mat` que almacenen unicamente las celdas que contengan calificaciones y no texto. Para ello, convertir el contenido de las columnas $\{2, \ldots, 10\}$ de `h01_cell` y `h02_cell` a variables del tipo `double` (precisión doble de 64 bits) a partir de `str2double()`. Nótese que esta modificación hace que las notas registradas como inasistencias *'F'* se conviertan en valores `NaN` (Not a Number).

Modificar las inasistencias a la nota 0 a partir de `isnan()`. Luego, determinar la nota mínima de laboratorios para cada uno de los alumnos no retirados del curso a partir de `min()` y almacenarlas en los vectores `labmin01_v` y `labmin02_v`.

Una vez calculadas las notas mínimas de laboratorio de los alumnos de cada horario, sumar las notas de las 5 sesiones, restar la mínima y obtener su promedio. Luego calcular la nota de presentaciones usando la rutina `mean()` y finalmente obtener las notas finales a partir de 4. Almacenarlas en la variables `final01_v` y `final02_v`.

e. A partir de `disp()`, mostrar el porcentage de aprobados y la nota final promedio de cada horario. El resultado deberá ser mostrado en la línea de comandos de la siguiente forma:

```
Nota promedio del horario h01: 12.0563. Porcentaje de aprobados: 90.625%
Nota promedio del horario h02: 11.9425. Porcentaje de aprobados: 83.3333%
```

f. De forma alternativa, calificar a todos los alumnos del curso basándose en una distribución normal a partir del sistema basado en letras: $\{\mathrm{A}+, \mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{F}\}$. Para ello, juntar las notas finales de ambos horarios en el vector `final_total_v` de 62 elementos de longitud. Luego, modificar la escala de notas de la siguiente forma:

$$\texttt{letra\_v(i)} = \begin{cases} \mathrm{A}+ & ,\text{si} \quad 2 \leq s_{\text{nota}}(i) \\ \mathrm{A} & ,\text{si} \quad 1 \leq s_{\text{nota}}(i) < 2 \\ \mathrm{B} & ,\text{si} \quad 0 \leq s_{\text{nota}}(i) < 1 \\ \mathrm{C} & ,\text{si} \quad -1 \leq s_{\text{nota}}(i) < 0 \\ \mathrm{D} & ,\text{si} \quad -2 \leq s_{\text{nota}}(i) \geq -1 \\ \mathrm{F} & ,\text{si} \quad -2 > s_{\text{nota}}(i) \end{cases},$$

Donde $s_{\text{nota}}(i)$ se define a partir de la media (μ) y desviación estandar (σ) del conjunto de notas[22]:

$$s_{\text{nota}}(i) = \frac{\texttt{final\_total\_v(i)} - \mu\{\texttt{final\_total\_v}\}}{\sigma\{\texttt{final\_total\_v}\}}$$

g. Finalmente, mostrar en línea de comandos el número de estudiantes con cada uno de los calificativos a partir de `sum()`, `disp()` y operaciones basadas en índices. El resultado debe mostrarse de la siguiente forma:

```
Calificaciones A+: 0
Calificaciones A: 6
Calificaciones B: 33
Calificaciones C: 17
Calificaciones D: 3
Calificaciones F: 3
```

[22] revisar las rutinas `mean()` y `std()`.