

Año 2021



Inteligencia Artificial

Trabajo Práctico N° 1

"Procesamiento Numérico"

Grupo: Neural

Alumnos:

Micaela del valle Castillo
Eduardo Matias Luna
Eduardo Matias Luna
Judith Graciela Patiño

LU:6271
LU: 4843
LU: 681
LU: 4520

Ing.Inf.
Ing. Inf.
Lic. Sist.
Ing. Inf.

1. Trabajo con matrices. Generar matrices con las características que se indican:

- a. Matriz de 6x7 de valores aleatorios enteros. Las primeras tres columnas impares en el intervalo (0,1), las restantes en el intervalo (-1,1).

```
function RANDOM_MATRIX = build_random_matrix(rows, cols, firstEvens)

A = [];
EVENS = 0;

for i=1:cols
    if rem(i,2) ~= 0 && EVENS <= firstEvens
        A = [A round(-1 + 2*rand(rows,1))];
        EVENS = EVENS + 1;
    else
        A = [A round(rand(rows,1))];
    end
end

RANDOM_MATRIX = A;

end
```

New to MATLAB? See resources for [Getting Started](#).

```
>> RANDOM_MATRIX = build_random_matrix(6, 7, 3)

RANDOM_MATRIX =

    1     0     1     1     0     1     0
    1     1     0     1     1     0     0
   -1     1     1     1     0     0     1
    1     1    -1     0     0     0    -1
    0     0     0     1     0     0     0
   -1     1     1     1    -1     1     0
```

https://github.com/matness-university/U1-procesamiento-numerico/blob/main/1_Trabajo_con_matrices/build_random_matrix.m

- b. Matriz A de 20x25 de valores enteros positivos. Extraer una matriz A1(3x8) desde la posición (5,5). Matriz A2 que resulte de la eliminación de las columnas 11 a 14 y las filas 9 a 13 de A.

```
function [A, SUB_MATRIX, MODIFIED_MATRIX] = get_sub_matrix(rows, cols)

A = round(rand(rows,cols)*100);

SUB_MATRIX = A(5:8,5:13);
```

```

MODIFIED_MATRIX = A;
MODIFIED_MATRIX(:,11:14) = [];
MODIFIED_MATRIX(9:13,:) = [];

end

% [A, SUB_MATRIX, MODIFIED_MATRIX] = get_sub_matrix(20, 25)

```

```

New to MATLAB? See resources for Getting Started.

>> [A, SUB_MATRIX, MODIFIED_MATRIX] = get_sub_matrix(20, 25)

A =
    59    89    59    93    25     3    85    86    35     6    24     8    40    72    21    29    45    17    26    30    74    12     6   100     9
    26    39    15    70    67    61    87    93    45    52     5    78     6    11     9    60    61    20    40    45    11    53    77    22    64
     4     77    20    58     8    36    27    98     5    34    44    91    78    12    77    96     6    32     7    42    68    33    67    65    18
    75    40    41    82    63     5    21    86    18    18     1    53    34    64    21    43    32    32    68    36    46    55    72    60     5
    24    81    75    88    66    49    56    79    66    21    90    11    61    33    39    69    77    22    40    56    21    40    64    39    72
    44    76    83    99    73    19    64    51    33    91    20    83    74    65    55    76    70    25    98    74    10    42    42    14    35
    69    38    79     0    89    12    42    18    90    68     9    34    10    75    23    43    13    89    40    42    82    18    39     3    66
    36    22    32    87    98    21    21    40    12    47    31    29    13    58    64    66    13    70    62    43    18    26    82    42    38
    74    79    53    61    77    15    95    13    99    91    46    75    55    74    48    11     9    56    15    12    16     2    32    18    63
    39    95     9    99    58    19     8     3    54    10    1     49    23    15    93     1    18    38     2    67    92    81    73     2
    68    33    11    53    93     4    11    94    71    75   100     5    89    73    78    19    42    21    16    29    89    65    79    37    91
    70    67    14    48    58    64    14    30   100    74    33    67    80    97    10    27    66     8    76    32    52    93    85    84    80
    44    44    68    80     2    28    17    30    29    56    30    60    73    87    29    80    72    91    87    65    70    16    51    73    75
     2     83    50    23    12    54    62    33    41    18     6    53     5     9    24    49    53    71    35    96    15    92    64    57    81
    33    77    19    50    86    70    57    47    46    60    30    73     7    37    53    77    11    56    69    94    95    79    95    18    38
    42    17    50    90    48    50     5    65    76    30     5    71     9    37     9    40    63    31    29    46    54    58    44    96    62
    27    86    15    57    84    54    93     3    82    13    51    78    80    69    41    27    13    17    53    24    68    44     6    27    58
    20    99     5    85    21    45    73    84    10    21    76    29    94    60    10     4    13    62    83    76     4    26    87    92    53
    82    51    85    74    55    12    74    56    18    89    63    69    68    79    11    67    10    99    60    76    81    75    63    22    28
    43    88    56    59    63    49     6    85    36     7     9    56    13    37    78    43    14    17    34    74    75    23    36    37    25

SUB_MATRIX =
    66    49    56    79    66    21    90    11    61
    73    19    64    51    33    91    20    83    74
    89    12    42    18    90    68     9    34    10
    98    21    21    40    12    47    31    29    13

MODIFIED_MATRIX =
    59    89    59    93    25     3    85    86    35     6    21    29    45    17    26    30    74    12     6   100     9
    26    39    15    70    67    61    87    93    45    52     9    60    61    20    40    45    11    53    77    22    64
     4     77    20    58     8    36    27    98     5    34    77    96     6    32     7    42    68    33    67    65    18
    75    40    41    82    63     5    21    86    18    21    43    32    32    68    36    46    55    72    60     5
    24    81    75    88    66    49    56    79    66    21    39    69    77    22    40    56    21    40    64    39    72
    44    76    83    99    73    19    64    51    33    91    55    76    70    25    98    74    10    42    42    14    35
    69    38    79     0    89    12    42    18    90    68    23    43    13    89    40    42    82    18    39     3    66
    36    22    32    87    98    21    21    40    12    47    64    66    13    70    62    43    18    26    82    42    38
    42    17    50    90    48    50     5    65    76    30     9    40    63    31    29    46    54    58    44    96    62
    27    86    15    57    84    54    93     3    82    13    41    27    13    17    53    24    68    44     6    27    58
    20    99     5    85    21    45    73    84    10    21    10     4    13    62    83    76     4    26    87    92    53
    82    51    85    74    55    12    74    56    18    89    11    67    10    99    60    76    81    75    63    22    28
    43    88    56    59    63    49     6    85    36     7    78    43    14    17    34    74    75    23    36    37    25

```

https://github.com/matnness-university/U1-procesamiento-numerico/blob/main/1_Trabajo_con_matrices/get_sub_matrix.m

- c. Matriz de 15x1 de números aleatorios enteros de dos dígitos. Determinar la posición y valor del menor y del mayor.

```

function [MIN, MIN_POS, MAX, MAX_POS] = get_max_min()

A = round(rand(15,1)*10)

MIN_POS = 1;
MAX_POS = 1;

for i=1:length(A)
    if A(i,1) < A(MIN_POS,1)
        MIN_POS = i;
    end
end

```

```
    if A(i,1) > A(MAX_POS,1)
        MAX_POS = i;
    end
end

MIN = A(MIN_POS,1);
MAX = A(MAX_POS,1);

end

% [MIN, MIN_POS, MAX, MAX_POS] = get_max_min()
```

```
>> [MIN, MIN_POS, MAX, MAX_POS] = get_max_min()

A =

     1
     7
    10
     4
    10
     3
     9
     5
     4
     2
     1
     3
     7
     8
     7

MIN =

     1

MIN_POS =

     1

MAX =

    10

MAX_POS =

     3

fx >>
```

https://github.com/matness-university/U1-procesamiento-numerico/blob/main/1_Trabajo_con_matrices/get_max_min.m

- d. Matriz aleatoria de 5x5, valores enteros en el intervalo (-25 ; 75). Ordenar por filas (orden creciente). Ordenar por columnas (orden decreciente).

```

function [ORDERED_MATRIX] = get_ordered_matrix()

A = round(-25 + 100*rand(5))

[rows, cols] = size(A);

for i=1:cols
    A(i,:) = sort(A(i,:));
end

for j=1:rows
    A(:,j) = sort(A(:,j));
end

ORDERED_MATRIX = A;

end

% get_ordered_matrix()

```

```

>> get_ordered_matrix()

A =

    -24    13    42    58   -15
     59    45    23    52    24
     67    48    37    68     -6
     52     -3     -1   -14    65
    -21     2     -7    -7   -15

ans =

    -24   -15     -7     -7     2
    -21   -15     -1    42    58
   -14     -3    13    52    59
     -6    24    45    52    65
     23    37    48    67    68

```

https://github.com/matnness-university/U1-procesamiento-numerico/blob/main/1_Trabajo_con_matrices/get_ordered_matrix.m

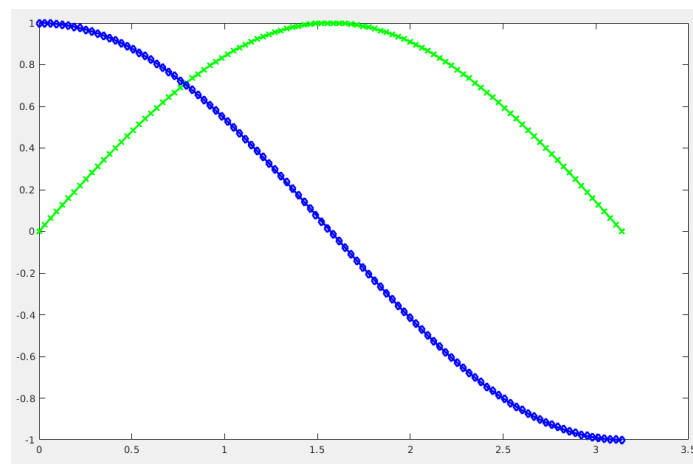
2. **Graficación.** Desde la línea de comandos ejecutar las sentencias necesarias para graficar las siguientes funciones:
 - a. Funciones seno y coseno en el intervalo $[0, 2\pi]$ con un mínimo de 100 puntos cada una, línea continua, color verde, grosor 2, marca x para el seno; color azul, grosor 2, marca rombo para el coseno. Graficar sobre el mismo sistema de ejes (ver comando hold on).

```
function [] = draw_charts()

x = linspace(0,pi,100);
y = sin(x);
p = plot(x,y);
p.Color = 'g';
p.LineWidth = 2;
p.Marker = 'x';
hold on
w = cos(x);
q = plot(x,w);
q.Color = 'b';
q.LineWidth = 2;
q.Marker = 'diamond';

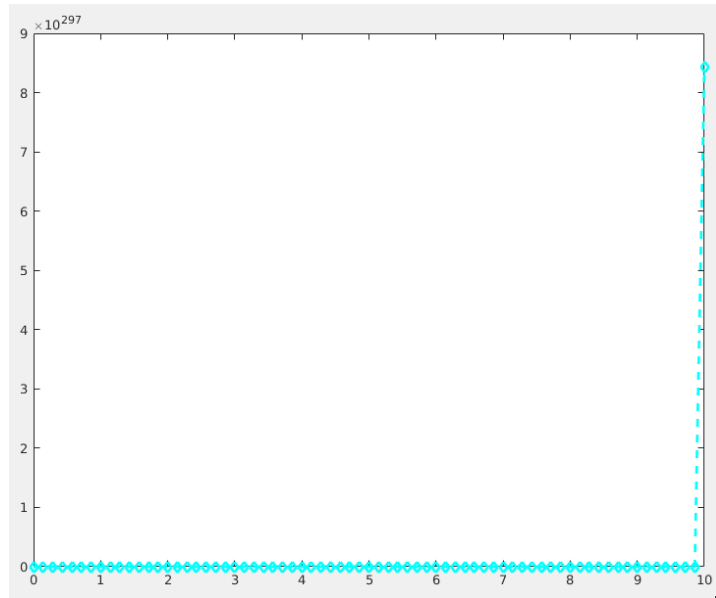
end
```

https://github.com/matnass-university/U1-procesamiento-numerico/blob/dde5d90b4d444036fdc7ff5a0cb0b6ef4604d8b6/2_Graficas/draw_charts.m



- b. $y_2 = e^{(x-5)^3/0.5}$ Intervalo (0 ; 10), línea de trazos, color cyan, grosor 2, marca rombo. Mínimo 70 puntos.

```
>> x = 0:10/70:10;
>> y = exp(((x-3).^3)/0.5);
>> p = plot(x,y);
>> p.LineStyle = '--';
>> p.Color = 'c';
>> p.LineWidth = 2;
>> p.Marker = 'diamond';
```



https://github.com/matnness-university/U1-procesamiento-numerico/blob/dde5d90b4d444036fdc7ff5a0cb0b6ef4604d8b6/2_Graficas/charts.txt

- c. Graficar la función definida por partes que se indica. Cada intervalo debe contener por lo menos 20 puntos. Cada sección debe ser de un color diferente.

$$f(x) = \begin{cases} -2,186 \cdot x - 12,864 & \text{si } -10 \leq x < -2 \\ 4,246 \cdot x & \text{si } -2 \leq x < 0 \\ 10 \cdot e^{(-0,05x - 0,5)} \cdot \text{sen}(0,03 \cdot x^2 + 0,7 \cdot x) & \text{si } 0 \leq x < 10 \end{cases}$$

```
function [] = draw_combined()

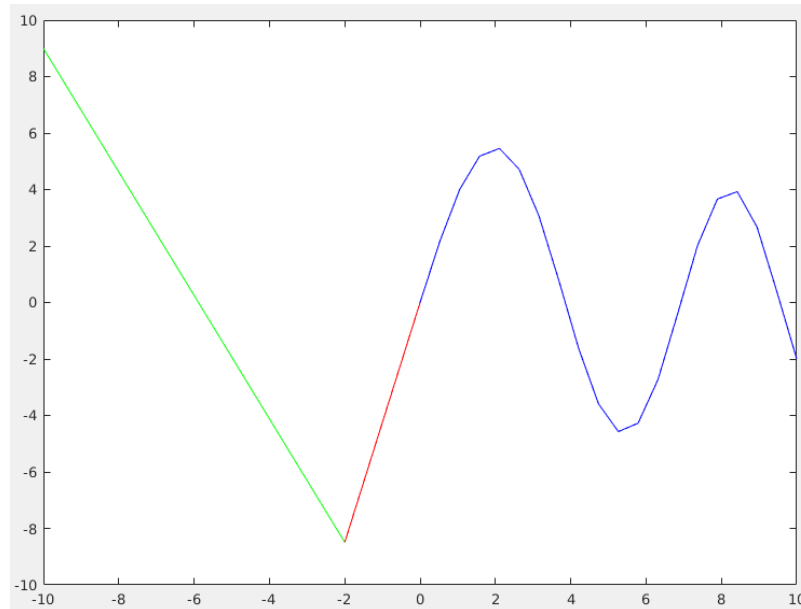
x = linspace(-10,-2,20);
f = -2.186*x - 12.864;

y = linspace(-2,0,20);
g = 4.246 * y;

z = linspace(0,10,20);
h = 10 * exp(-0.05 * z - 0.5) .* sin(0.03 * z.^2 + 0.7 * z);

p = plot(x,f);
p.Color = 'g';
hold on
q = plot(y,g);
q.Color = 'r';
hold on
r = plot(z,h);
r.Color = 'b';

end
```



https://github.com/matness-university/U1-procesamiento-numerico/blob/dde5d90b4d444036fdc7ff5a0cb0b6ef4604d8b6/2_Graficas/draw_combined.m

3. Series. Generar las series que se indican:

- a. Generar una serie de 33 números aleatorios enteros de 3 cifras, en el intervalo [100;200].

```
% r = a + (b-a) .*rand(N,1)
% get_serie(33,100,200)

function [SERIE] = get_serie(total, leftValue, rightValue)

SERIE = round(leftValue + (rightValue - leftValue) .*rand(1,total));

end
```

https://github.com/matness-university/U1-procesamiento-numerico/blob/dde5d90b4d444036fdc7ff5a0cb0b6ef4604d8b6/3_Series/get_serie.m

- b. Generar una serie S1 en el intervalo [0 ; 2] con un intervalo de $\pi/3.3$ (usar el operador ':'). Generar una serie S2 en el mismo intervalo que contenga 7 elementos (usar la función linspace). Comparar ambas series y explicar sus diferencias.

```
function [FIRST_SERIE, SECOND_SERIE] = get_series()

FIRST_SERIE = 0:pi/3.3:2*pi;
SECOND_SERIE = linspace(0,2*pi,7);
```



```
% En FIRST_SERI los valores son equidistantes, mientras que en
SECOND_SERIE
% pueden no serlo

end
```

https://github.com/matnness-university/U1-procesamiento-numerico/blob/dde5d90b4d444036fdc7ff5a0cb0b6ef4604d8b6/3_Series/get_series.m

c. Explicar cuál es el problema al intentar generar las siguientes series:

>> [10 : -2 : 25]

La cota inferior debería ser mayor a la superior, en este caso [25:-2:10]

>> [-5 : 11 : 9]

No hay inconvenientes al ejecutar esta serie

>> [42 : 2.5 : 23]

Es necesario agregar un paso negativo para poder obtener una serie decreciente

https://github.com/matnness-university/U1-procesamiento-numerico/blob/dde5d90b4d444036fdc7ff5a0cb0b6ef4604d8b6/3_Series/pregutnas.txt

4. Escribir funciones en línea de comando. Detectar e indicar cuál es el problema que ocurre al intentar escribir en Matlab las siguientes funciones, solucionarlo y escribirlas adecuadamente:

a. >> P1 = a * x^2 + b * x + c

Siendo a,b,c constantes definidas, es necesario el operador '.' al elevar una matriz a una potencia

P1 = a * x.^2 + b * x + c

b. >> F(A,B,C) = 3 + 9^2 + pi

Las variables A,B,C no están definidas y tampoco inciden en el resultado de la función, es una constante, en todo caso podemos escribir esta función como

F = 3 + 9^2 + pi

c. >> a=5; b=21; x=[4 5 6]; f1 = a*x+b*x^2

Es necesario el operador '.' al elevar una matriz a una potencia

a=5; b=21; x=[4 5 6]; f1 = a*x+b*x.^2

d. >> m=ones(2); n=magic(2); x=[7 8 9]; g1=m^2*x+x^*n

^* No es una correcta implementación de estos operadores. Por otro lado, las matriz 'm' tiene dimensiones 2x2, incompatibles para una multiplicación matricial con el vector x de 3 elementos

e. ¿Para qué sirve el operador ' .' y el operador ' ; ' ?

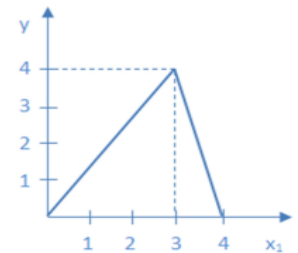
El operador '.' permite una operación escalar en una matriz

El operador ';' evita mostrar el resultado del comando por pantalla

https://github.com/matnness-university/U1-procesamiento-numerico/blob/dde5d90b4d444036fdc7ff5a0cb0b6ef4604d8b6/4_Comandos/preguntas.txt

5. Cambio de escala (reescalado).

- a. Considerando la gráfica adjunta, escribir sus ecuaciones. Con ellas, generar una secuencia de 40 puntos para cada eje. Reescalar la secuencia de ordenadas (y) al intervalo (0,1). Graficar las secuencias (original y reescalada) sobre el mismo sistema de ejes. Calcular el centro de gravedad de ambas gráficas.



```
function [CX_1,CY_1,CX_2,CY_2] = scales()

x_1 = [linspace(0,3,30) linspace(3.1,4,10)];
y_1 = [];

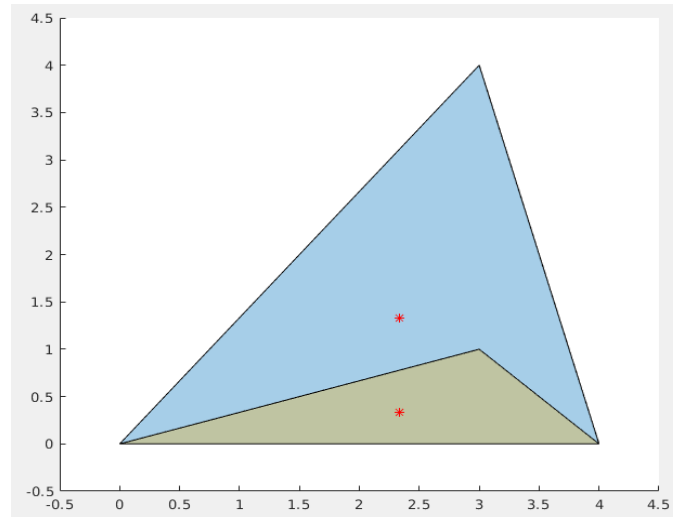
for i=1:length(x_1)
    if x_1(i) < 3
        y_1(i) = 4/3*x_1(i);
    else
        y_1(i) = 16-4*x_1(i);
    end
end

y_2 = mapminmax(y_1,0,1);

polyin_1 = polyshape(x_1,y_1);
[CX_1,CY_1] = centroid(polyin_1);
plot(polyin_1);
hold on;
plot(CX_1,CY_1,'r*');
hold on;

polyin_2 = polyshape(x_1,y_2);
[CX_2,CY_2] = centroid(polyin_2);
plot(polyin_2);
hold on;
plot(CX_2,CY_2,'r*');
hold on;

end
```



https://github.com/matnness-university/U1-procesamiento-numerico/blob/dde5d90b4d444036fdc7ff5a0cb0b6ef4604d8b6/5_Escalas/scales.m

- b. La ecuación siguiente representa a una función wavelet Morlet. Graficarla en el intervalo $(-10,10)$ con un mínimo de 100 puntos.

```
function [] = wavelet()

x = linspace(-10,10,100);

y_1 = cos(3*x) .* exp(-(x.^2)/2);
y_2 = mapminmax(y_1,-1,0);

plot(x,y_1);
hold on;
plot(x,y_2);

end
```

https://github.com/matnness-university/U1-procesamiento-numerico/blob/dde5d90b4d444036fdc7ff5a0cb0b6ef4604d8b6/5_Escalas/wavelet.m

6. Ruido. Sobre la secuencia temporal generada por la ecuación:

$$y = 10 * \sin(2*t) + 5$$

- a. Obtener 50 puntos para que t pertenezca a $[0,120^\circ]$. Verificar en qué unidades trabaja $\sin()$.

```
function [] = get_points()

x = linspace(0,2/3*pi,50);
```

```
y = 10 * sin(2*x) + 5;

end
```

Sin trabajar en radianes.

https://github.com/matness-university/U1-procesamiento-numerico/blob/dde5d90b4d444036fdc7ff5a0cb0b6ef4604d8b6/6_Ruido/get_points.m

- b. Agregar, a la secuencia, ruido blanco con una amplitud máxima (positiva o negativa) del 10% de la senoide de base (quitando la componente de continua → 5).**

```
function [] = get_noice()

x_1 = linspace(0,2/3*pi,50);
y_1 = 10 * sin(2*x_1) + 5;

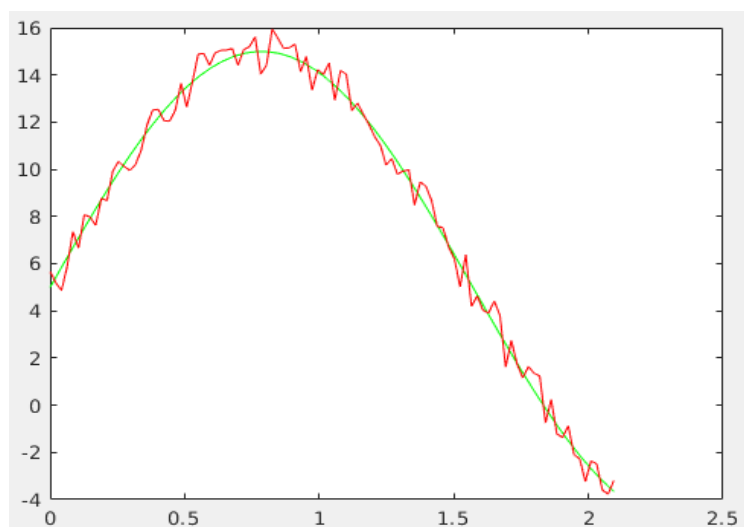
x_2 = linspace(0,2/3*pi,100);
y_2 = 10 * sin(2*x_2) + 5 - 1 + 2 * rand(1,100);

plot(x_1, y_1, 'g');
hold on;
plot(x_2, y_2, 'r');

end
```

https://github.com/matness-university/U1-procesamiento-numerico/blob/dde5d90b4d444036fdc7ff5a0cb0b6ef4604d8b6/6_Ruido/get_noice.m

- c. Graficar la señal sin ruido en rojo y superponer la señal con ruido en azul.**



- d. Calcular para ambas secuencias, la media (μ) y la desviación estándar (σ).

```
function [] = get_noise()

x_1 = linspace(0,2/3*pi,50);
y_1 = 10 * sin(2*x_1) + 5;

MEAN_1 = mean(y_1);
STANDARD_1 = std(y_1);

x_2 = linspace(0,2/3*pi,100);
y_2 = 10 * sin(2*x_2) + 5 - 1 + 2 * rand(1,100);

MEAN_2 = mean(y_2);
STANDARD_2 = std(y_2);

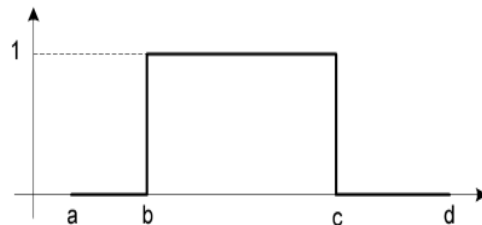
plot(x_1, y_1, 'g');
hold on;
plot(x_2, y_2, 'r');

end
```

https://github.com/matness-university/U1-procesamiento-numerico/blob/main/6_Ruido/get_noise.m

7. Scripts

- a. Escribir un script que dibuje un pulso como el de la figura. Los puntos (a, b, c, d) deben ser solicitados al usuario. Los flancos deben verse verticales.



```
function [] = draw()

prompt = 'Ingresse a: ';
a = input(prompt);

prompt = 'Ingresse b: ';
b = input(prompt);

prompt = 'Ingresse c: ';
c = input(prompt);

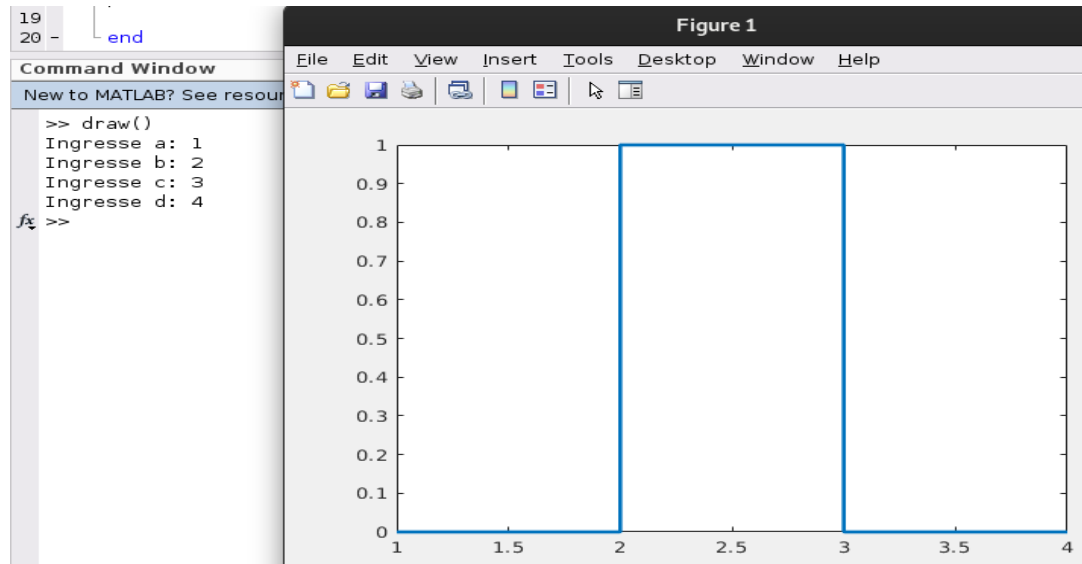
prompt = 'Ingresse d: ';
d = input(prompt);
```

```

x = [a,b,b,c,c,d];
y = [0,0,1,1,0,0];
p = plot(x,y);
p.LineWidth = 2;

end

```



https://github.com/matnass-university/U1-procesamiento-numerico/blob/dde5d90b4d444036fdc7ff5a0cb0b6ef4604d8b6/7_Scripts/draw.m

- b. Escribir un script que genere una matriz identidad de 10x10. Luego, el script debe rellenar la matriz con valores aleatorios enteros del intervalo [2 ; 9], excepto en las celdas donde están los '1'. Orientación: revisar las funciones `eye()` y `randi()` .

```

function [X] = create_matrix()

x_2 = randi([2,9],10);

for i=1:10
    for j=1:10
        if(i~=j)
            X(i,j) = x_2(i,j);
        else
            X(i,j) = 1;
        end
    end
end
end

```

```
end
```

```
>> create_matrix()
ans =
     1     3     3     4     4     4     4     2     7     2
     2     1     2     6     5     5     2     4     4     2
     7     6     1     5     5     2     8     3     8     2
     6     3     3     1     8     7     9     3     9     9
     4     2     2     5     1     7     8     4     8     6
     6     2     3     3     2     1     8     6     4     8
     4     8     2     8     2     9     1     2     7     6
     3     7     3     7     3     2     3     1     2     7
     4     2     2     2     6     8     5     5     1     4
     2     7     2     5     9     5     4     5     2     1
```

https://github.com/matnass-university/U1-procesamiento-numerico/blob/dde5d90b4d444036fdc7ff5a0cb0b6ef4604d8b6/7_Scripts/create_matrix.m

- c. Escribir un script para ejecutar en forma automática el problema 3.a) anterior. La cantidad de números de la serie debe ser introducida por el usuario (función `input`), lo mismo que la cantidad de cifras de los números y el intervalo de la serie.

Prob. 3.a) Generar una serie de 33 números aleatorios enteros de 3 cifras, en el intervalo [100;200].

```
function [SERIE] = get_serie()

% r = a + (b-a).*(rand(N,1))

prompt = 'Ingresse max: ';
max = input(prompt);

prompt = 'Ingresse min: ';
min = input(prompt);

prompt = 'Ingresse cantidad de números: ';
elements = input(prompt);

SERIE = round(min + (max-min-1).*(rand(1,elements)));

end
```

```

>> get_serie()
Ingresse max: 200
Ingresse min: 100
Ingresse cantidad de números: 33

ans =

Columns 1 through 26
    171    150    147    106    168    104    107    152    110    181    181    172    115    165    151    196    164    179    145    143    182    108    113    117    139    182

Columns 27 through 33
    180    106    140    152    141    165    162

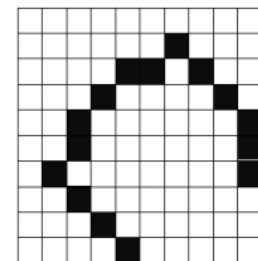
fx >> |

```

https://github.com/matnness-university/U1-procesamiento-numerico/blob/dde5d90b4d444036fdc7ff5a0cb0b6ef4604d8b6/7_Scripts/get_serie.m

8. Scripts 2. Dada la siguiente matriz booleana, diseñar un script que:

- Genere 15 matrices booleanas, del mismo tamaño, conteniendo cada una 15 “unos” dispuestos en forma aleatoria.**
- Compare las 15 matrices con la matriz de muestra e identifique las matrices de mayor y menor similitud.**
- Por cada comparación realizada genere un índice de similitud ($IS \in R$) entre 0 y 1 (un valor 1 significa que las matrices son iguales). Orientación: investigar el índice de Jaccard.**



Blanco = '0'

```

function [RANDOM_MATRIX] = create_pattern_matrix()

PATTERN_INDEXES = [17 25 26 28 34 39 43 50 53 60 62 70 73 84 95];
ZEROS = zeros(1,100);

for i=1:length(PATTERN_INDEXES)
    ZEROS(PATTERN_INDEXES(i)) = 1;
end

RANDOM_MATRIX = reshape(ZEROS,[10,10]);

end

```

https://github.com/matnness-university/U1-procesamiento-numerico/blob/dde5d90b4d444036fdc7ff5a0cb0b6ef4604d8b6/8_Scripts_2/create_pattern_matrix.m

```

function [RANDOM_MATRIX] = create_random_matrix()

RANDOM_INDEXES = randperm(100,15);
ZEROS = zeros(1,100);

```



```

for i=1:length(RANDOM_INDEXES)
    ZEROS(RANDOM_INDEXES(i)) = 1;
end

RANDOM_MATRIX = reshape(ZEROS,[10,10]);

end

```

https://github.com/matnness-university/U1-procesamiento-numerico/blob/dde5d90b4d444036fdc7ff5a0cb0b6ef4604d8b6/8_Scripts_2/create_random_matrix.m

```

function [PATTERN, RANDOM, MATCH_RATES] = compare_matrix()

PATTERN_MATRIX = create_pattern_matrix;

% Generamos 15 matrices con un patrón de 15 unos random
for i=1:15
    RANDOM_MATRIXES{i} = create_random_matrix();
end

% Sumamos cada matriz con el patrón para asignar un peso a cada acierto
for i=1:15
    A = PATTERN_MATRIX + RANDOM_MATRIXES{i};

    % quitamos las celdas que no tuvieron aciertos
    A(A==1) = 0;

    % limpiamos los valores sumados en los aciertos
    COMPARED_MATRIXES{i} = A/2;
end

for i=1:15
    % sumamos por separado los aciertos en cada matriz de comparación
    MATCH_RATES(i) = sum(COMPARED_MATRIXES{1,i}, 'all') /
sum(PATTERN_MATRIX, 'all');

PATTERN = PATTERN_MATRIX;
RANDOM = RANDOM_MATRIXES;

end

```

https://github.com/matnness-university/U1-procesamiento-numerico/blob/dde5d90b4d444036fdc7ff5a0cb0b6ef4604d8b6/8_Scripts_2/compare_matrix.m

Patrón

Editor - compare_matrix.m

Variables - PATTERN

10x10 double

	1	2	3	4	5	6	7	8	9	10	11
1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	1	0	0	0
3	0	0	0	0	1	1	0	1	0	0	0
4	0	0	0	1	0	0	0	0	1	0	0
5	0	0	1	0	0	0	0	0	0	1	0
6	0	0	1	0	0	0	0	0	0	0	0
7	0	1	0	0	0	0	0	0	0	0	0
8	0	0	1	0	0	0	0	0	0	0	0
9	0	0	0	1	0	0	0	0	0	0	0
10	0	0	0	0	1	1	1	0	0	0	0
11											

Casos con 0% de similitud

```
>> [PATTERN, RANDOM, MATCH_RATES] = compare_matrix()

PATTERN =

0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 1 1 0 1 0 0 0
0 0 0 1 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0 0 0 1
0 0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 1 1 1 0 0 0 0

RANDOM =

1x15 cell array
Columns 1 through 8
{10x10 double} {10x10 double} {10x10 double} {10x10 double} {10x10 double} {10x10 double} {10x10 double} {10x10 double}
Columns 9 through 15
{10x10 double} {10x10 double} {10x10 double} {10x10 double} {10x10 double} {10x10 double} {10x10 double}

MATCH_RATES =

0 0.0667 0.2667 0.1333 0.2000 0.2000 0.1333 0.2667 0.2667 0.1333 0.1333 0.1333 0.2000 0 0.1333
```

Primer valor del vector de matches (0%)

Editor - compare_matrix.m

Variables - RANDOM{1, 1}

RANDOM{1, 1}

	1	2	3	4	5	6	7	8	9	10	11
1	1	0	0	0	0	0	0	0	1	0	
2	0	1	0	0	0	0	0	0	0	0	
3	0	1	0	0	0	0	1	0	0	0	
4	0	0	0	0	0	0	0	1	0	0	
5	0	0	0	0	1	1	0	0	0	0	
6	0	0	0	0	1	0	0	0	0	0	
7	0	0	0	0	1	0	1	1	0	0	
8	1	0	0	0	0	0	0	0	0	0	
9	0	0	0	0	0	0	0	0	0	1	
10	0	1	0	0	0	0	0	0	0	0	
11											

Penúltimo valor en el vector de matches (0%)

Editor - compare_matrix.m

Variables - RANDOM{1, 14}

RANDOM{1, 14}

	1	2	3	4	5	6	7	8	9	10	11
1	1	1	0	1	0	0	0	1	0	0	
2	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	1	0	0	0	0	1	0	
4	0	0	0	0	0	0	1	0	0	0	
5	0	0	0	0	0	0	1	0	0	0	
6	0	0	0	0	0	0	0	0	0	0	
7	0	0	1	1	0	0	0	0	0	0	
8	0	1	0	0	1	0	0	1	0	0	
9	0	0	0	0	0	0	0	0	0	0	
10	0	0	0	0	0	0	0	1	1	0	

Casos con 40% de similitud

New to MATLAB? See resources for [Getting Started](#).

```
>> [PATTERN, RANDOM, MATCH_RATES] = compare_matrix()

PATTERN =

    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    1    0    0    0    0
    0    0    0    0    1    1    0    1    0    0    0
    0    0    0    1    0    0    0    0    0    1    0
    0    0    1    0    0    0    0    0    0    0    1
    0    0    1    0    0    0    0    0    0    0    0
    0    1    0    0    0    0    0    0    0    0    0
    0    0    1    0    0    0    0    0    0    0    0
    0    0    0    1    0    0    0    0    0    0    0
    0    0    0    0    1    0    0    0    0    0    0
    0    0    0    0    1    1    1    0    0    0    0

RANDOM =
1x15 cell array
Columns 1 through 8
    {10x10 double}    {10x10 double}    {10x10 double}    {10x10 double}    {10x10 double}    {10x10 double}    {10x10 double}    {10x10 double}
Columns 9 through 15
    {10x10 double}    {10x10 double}    {10x10 double}    {10x10 double}    {10x10 double}    {10x10 double}    {10x10 double}

MATCH_RATES =
    0.1333    0.4000    0.0667    0.1333    0.1333    0.2667    0.2000    0.0667    0.2000    0.0667    0.0667    0.2667    0.1333    0.0667    0
```

Segundo valor en el vector de matches (40%)

Editor - compare_matrix.m

Variables - RANDOM{1, 2}

RANDOM{1, 2}

	1	2	3	4	5	6	7	8	9	10	11
1	0	0	0	0	0	0	0	0	0	0	1
2	0	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	1	0
5	1	0	0	0	0	0	0	0	0	0	0
6	0	0	1	0	0	0	0	0	0	0	0
7	0	1	0	0	0	0	0	1	1	1	1
8	0	0	0	0	0	1	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	1	1	1	0	1	1	1

- d. Rediseñe el script para generar 15 matrices con escala de grises. Conviértelas a booleanas con 1 si la celda posee un gris $\geq 75\%$, 0 si la celda contiene un gris $<15\%$ y un valor aleatorio para los restantes casos.

```
function [RANDOM_MATRIX] = create_grey_based_random_matrix()

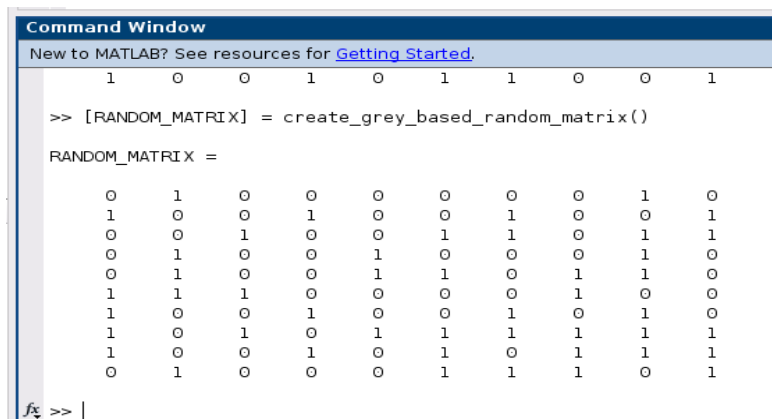
GREY_SCALE_SERIE = rand(1,100);

for i=1:length(GREY_SCALE_SERIE)
    if (GREY_SCALE_SERIE(i) > 0.75)
        GREY_SCALE_SERIE(i) = 1;
    elseif (GREY_SCALE_SERIE(i) < 0.15)
        GREY_SCALE_SERIE(i) = 0;
    else
        GREY_SCALE_SERIE(i) = round(rand());
    end
end

RANDOM_MATRIX = reshape(GREY_SCALE_SERIE, [10,10]);

end
```

https://github.com/matnness-university/U1-procesamiento-numerico/blob/dde5d90b4d444036fdc7ff5a0cb0b6ef4604d8b6/8_Scripts_2/create_grey_based_random_matrix.m



Command Window

New to MATLAB? See resources for [Getting Started](#).

```
>> [RANDOM_MATRIX] = create_grey_based_random_matrix()

RANDOM_MATRIX =

    0    1    0    0    0    0    0    0    1    0
    1    0    0    1    0    0    1    0    0    1
    0    0    1    0    0    1    1    0    1    1
    0    1    0    0    1    0    0    0    1    0
    0    1    0    0    1    1    0    1    1    0
    1    1    1    0    0    0    0    1    0    0
    1    0    0    1    0    1    1    0    1    0
    1    0    1    0    1    1    1    1    1    1
    1    0    0    1    0    1    0    1    1    1
    0    1    0    0    0    1    1    1    0    1
```

9. Función 1. Identificar qué hace la función que sigue. Previamente deben ser encontrados y corregidos dos errores en el código.

```
function p = diagonal(matriz,valor)
FilCol=size(matriz);
if (valor~=0)&&(valor~=1)
    error('El 2do parámetro debe ser 0 o 1')
end
if FilCol(1)~=FilCol(2)
    error('La matriz debe ser cuadrada')
end
for i=1:FilCol(1)
    for j=1:FilCol(2)
        if i<j || i>j
            matriz(i,j)=valor;
        end
    end
end
p=matriz;
```

La función propuesta recibe una matriz cuadrada junto a un valor 0 ó 1 y devuelve una matriz de iguales dimensiones con la diagonal principal igual a la de la matriz original y el resto de valores seteados a 0 ó 1, según la elección del usuario. Se corrigen dos 'end' faltantes en la función

```
function p = diagonal(matriz,valor)

FilCol=size(matriz);

if (valor~=0)&&(valor~=1)
    error('El 2do parámetro debe ser 0 o 1')
end

if FilCol(1)~=FilCol(2)
    error('La matriz debe ser cuadrada')
end

for i=1:FilCol(1)
    for j=1:FilCol(2)
        if i<j || i>j
            matriz(i,j)=valor;
        end
    end
end

p=matriz;

end
```

https://github.com/matnness-university/U1-procesamiento-numerico/blob/dde5d90b4d444036fdc7ff5a0cb0b6ef4604d8b6/9_Funcion/diagonal.m

New to MATLAB? See resources for [Getting Star](#)

```
>> a
a =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9

>> diagonal(a,0)
ans =
    17     0     0     0     0
     0     5     0     0     0
     0     0    13     0     0
     0     0     0    21     0
     0     0     0     0     9
```

10. Función 2. Escribir una función con las características indicadas:

- Debe generar una matriz A (dimensión 15x15) de números enteros aleatorios en el rango [-1, 1].
- Debe encontrar la primera ocurrencia de una submatriz B 3x2 , dada por el usuario como argumento de la función, dentro de la matriz A. En caso de no encontrar la matriz la submatriz B, debe encontrar la primera ocurrencia de 2x2 que no contenga números negativos.
- La función devolverá la posición (fila y columna) de la matriz encontrada. En caso contrario generará un mensaje de error indicando tal situación.
- La función debe verificar que la matriz ingresada sea de 3x2 y que sus valores estén en el rango indicado para A. Caso contrario, se emitirá un mensaje de que no encontró la matriz ingresada y continuará buscando la matriz binaria de 2x2.

```
function [MATRIX] = create_matrix()

MATRIX = round(-1 + 2*rand(15));

end
```

https://github.com/matness-university/U1-procesamiento-numerico/blob/dde5d90b4d444036fdc7ff5a0cb0b6ef4604d8b6/10_Funcion/create_matrix.m

```
>> create_matrix()
ans =
-1    0    1    0    1    0   -1    0    1    1    0   -1   -1    0   -1
 0   -1    0    0    0    1    0   -1   -1    0    0    0    0   -1
 0    1    0    0   -1    0    0   -1    0    0    1    1    0    1   -1
 1    1    1   -1   -1    0   -1    1    0    0    0   -1    1    0   -1
 1    0   -1   -1    0    1    0   -1    1    0   -1   -1   -1    0    0
-1    1    0    0    1    0   -1    1    0   -1    1   -1    1    1    0
 1    0    0    1    0    1    1    0    1   -1    0    0    1    1    1
 1    0    1    0    1    1    0    1   -1   -1   -1    0    0    0    0
 0    0    1    0   -1    0    0   -1    0   -1    0    0    0    0   -1
 1   -1   -1   -1    1    0   -1    0   -1   -1   -1    0    0    0    1
-1    0    0    1    0   -1    0   -1   -1    0   -1    0    0   -1    1
 0   -1    0    0   -1   -1    0    1    1   -1    1    0    0    0    0
 1    0    0    0    0    0    0   -1    0    1    1    0    0    0   -1
 1   -1    0    0    0    1    0    1    0    1    0   -1    1   -1    0
 1   -1    1    1    0    1    0    1   -1    0   -1    0    1    1    0
```

```
function [SUB_MATRIX, INDEXES] = sub_matrix(A, B)

[A_rows, A_cols] = size(A);
[B_rows, B_cols] = size(B);

if (A_rows < B_rows || A_cols < B_cols)
    error('La matriz a buscar tiene una dimensi3n mayor')
end

if (B_rows ~= 3 || B_cols ~= 2)
    error('La matriz a buscar tiene que ser de dimensi3n 3x2')
end

INDEXES = [];

if(B_rows == 3 || B_cols == 2)
    for i=1:A_rows-2
        for j=1:A_cols-1
            if isequal(A(i:i+2,j:j+1), B)
                SUB_MATRIX = A(i:i+2,j:j+1);
                INDEXES = [i,j];
                break;
            end
        end
    end
end

if size(INDEXES) == 0
    for i=1:A_rows-1
        for j=1:A_cols-1
            C = A(i:i+1,j:j+1);
            if min(min(C)) >= 0
                SUB_MATRIX = C;
                INDEXES = [i,j];
            end
        end
    end
end
```

```

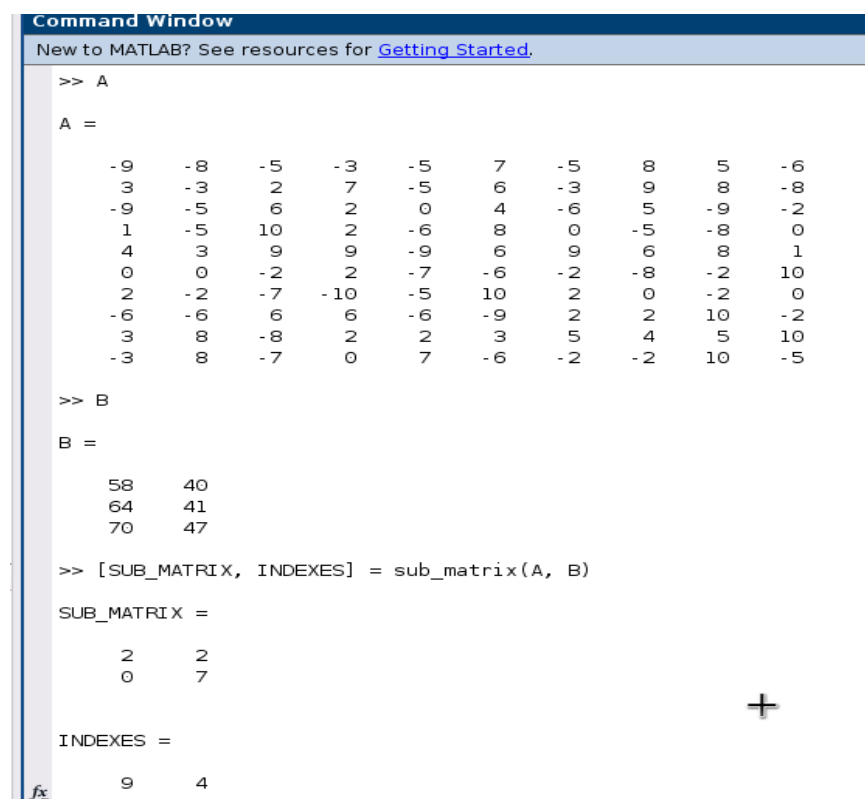
        break;
    end
end
end
end

if size(INDEXES) == 0
    error('no se ha encontrado la submatriz')
end

end

```

https://github.com/matnness-university/U1-procesamiento-numerico/blob/dde5d90b4d444036fdc7ff5a0cb0b6ef4604d8b6/10_Funcion/sub_matrix.m



Command Window

New to MATLAB? See resources for [Getting Started](#).

```

>> A

A =

    -9    -8    -5    -3    -5     7    -5     8     5    -6
     3    -3     2     7    -5     6    -3     9     8    -8
    -9    -5     6     2     0     4    -6     5    -9    -2
     1    -5    10     2    -6     8     0    -5    -8     0
     4     3     9     9    -9     6     9     6     8     1
     0     0    -2     2    -7    -6    -2    -8    -2    10
     2    -2    -7   -10    -5    10     2     0    -2     0
    -6    -6     6     6    -6    -9     2     2    10    -2
     3     8    -8     2     2     3     5     4     5    10
    -3     8    -7     0     7    -6    -2    -2    10    -5

>> B

B =

    58    40
    64    41
    70    47

>> [SUB_MATRIX, INDEXES] = sub_matrix(A, B)

SUB_MATRIX =

     2     2
     0     7

INDEXES =

     9     4

```