Introducción a MATLAB

PROCESAMIENTO DE SEÑALES E IMÁGENES

Profesores:

- Luis Corral
- Cristobal Loyola

Ayudante: Clemente Aguilar

Introducción a MATLAB

En MATALB representamos las variables numéricas como vectores y matrices. Cuando omitimos el punto y coma al final de una línea de código, su valor puede ser visto en el panel lateral. Podemos utilizar los operadores de suma, resta multiplicación, división y exponenciación (+,-,*,/,^). Un punto antes del operador indica operación elemento a elemento.

```
clearvars         % Borra todas las variables de la memoria.

a = [1 2 3 4 5 6] % Vector fila 1x6.
```

```
a = 1×6
     1     2     3     4     5     6
```

```
b = a'            % La transpuesta genera un vector
```

```
b = 6×1
     1
     2
     3
     4
     5
     6
```

```
                  % columna 6x1.
a_2 = [1 0; 0 1]  % Matrix identidad de 2x2.
```

```
a_2 = 2×2
     1     0
     0     1
```

```
a_3 = (((a+3)*2)/4).^2 % Operaciones matemáticas.
```

```
a_3 = 1×6
    4.0000    6.2500    9.0000   12.2500   16.0000   20.2500
```

Las funciones generadoras permiten crear datos enteros y de punto flotante de manera simple. Existen variadas formas de crear matrices y vectores, las cuales pueden ser consultadas en la documentación MATLAB Help.

```
c_1 = 0:1:5             % Vectores espaciados linealmente
```

```
c_1 = 1×6
     0     1     2     3     4     5
```

```
                        % inicio:distancia:fin o,
c_2 = linspace(0,5,6) % linspace(inicio,fin,largo).
```

```
c_2 = 1×6
     0     1     2     3     4     5
```

```
d = zeros(2,3)          % Matriz de zeros 2x3
```

```
d = 2×3
     0     0     0
     0     0     0
```

Para cambiar las dimensiones y valores, utilizamos concatenaciones, slicing, reshape, o permute (más la ya mencionada transpuesta) según corresponda:

```
d_2 = permute(d,[2,1]) % Matriz de zeros 3x2.
```

```
d_2 = 3×2
     0     0
     0     0
     0     0
```

```
c_1(1,3:end) = 0        % Se cambian los valores desde la
```

```
c_1 = 1×6
     0     1     0     0     0     0
```

```
                        % posición 3 hasta el final por 0.
c_3 = [[c_1;c_2] d]     % Matriz de 2x9
```

```
c_3 = 2×9
     0     1     0     0     0     0     0     0     0
     0     1     2     3     4     5     0     0     0
```

```
c_4 = [1 2;3 4]         % Matriz de 2x2
```

```
c_4 = 2×2
     1     2
     3     4
```

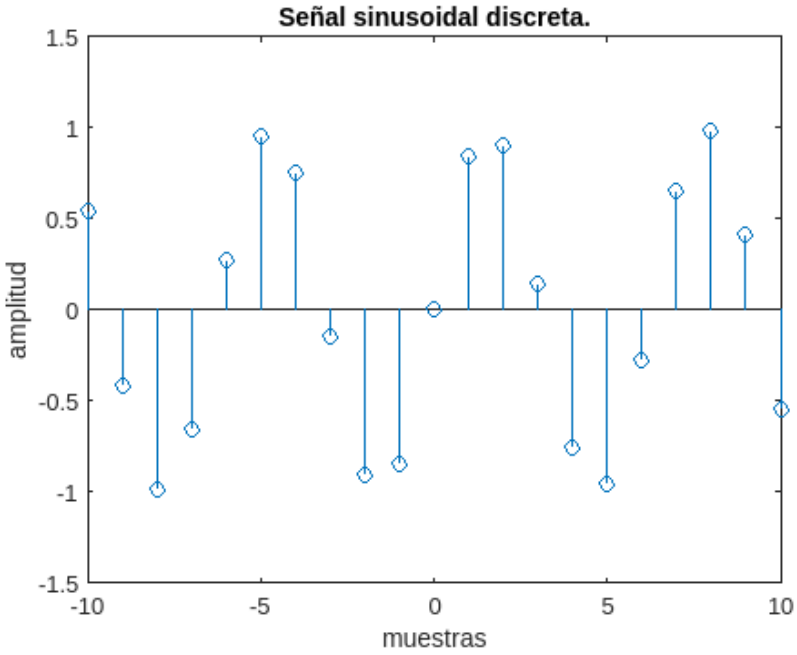
```
d = reshape(d,[1,6])    % Vector de zeros 1x6
```

```
d = 1×6
     0     0     0     0     0     0
```

Señales discretas

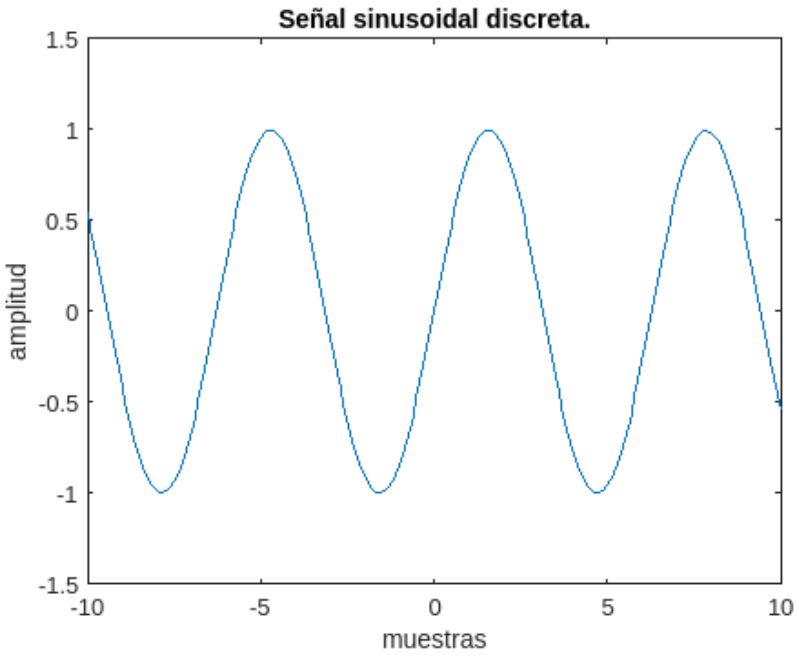
La representación de señales discretas se puede realizar utilizando la función stem(eje_x, eje_y) para generar gráficos. Las funciones title, xlabel, ylabel e ylim nos permiten dar formato al gráfico. Las funciones trigonométricas funcionan sobre el arreglo directamente.

```
clearvars

n = -10:1:10;        % Utilizamos n para señales discretas
x_n = sin(n);        % (21 muestras).
figure               % Crea una figura para el gráfico.
stem(n,x_n)
title('Señal sinusoidal discreta.')
xlabel('muestras')   % Etiqueta del eje x.
ylabel('amplitud')   % Etiqueta del eje y.
ylim([-1.5 1.5])     % Límites del eje y.
```



Para señales de mayor cantidad de muestras, la función plot(eje_x, eje_y) nos entrega un gráfico de líneas entre cada muestra. Ambas señales son discretas, pero utilizamos plot para mejorar la visualización.

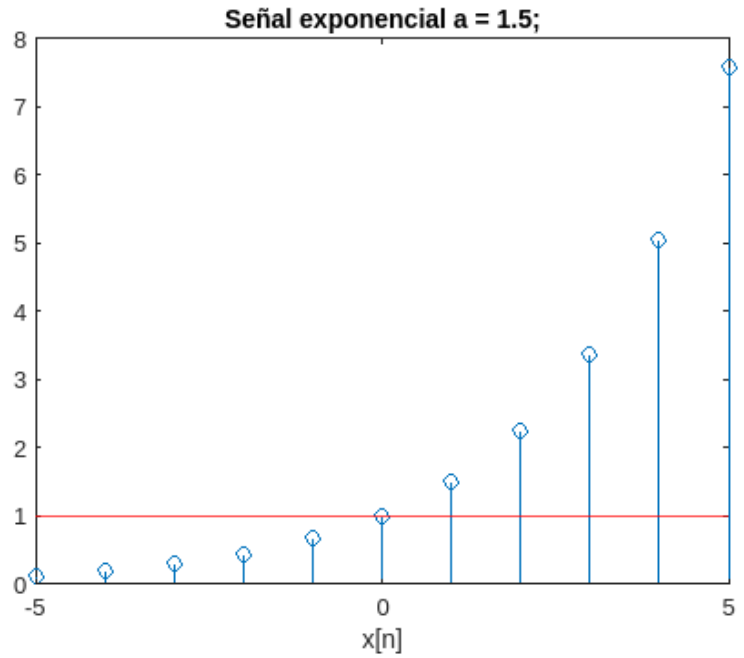
```
n = -10:0.1:10;      % 201 muestras.
x_n = sin(n);
figure
plot(n,x_n)
title('Señal sinusoidal discreta.')
xlabel('muestras')
ylabel('amplitud')
ylim([-1.5 1.5])
```



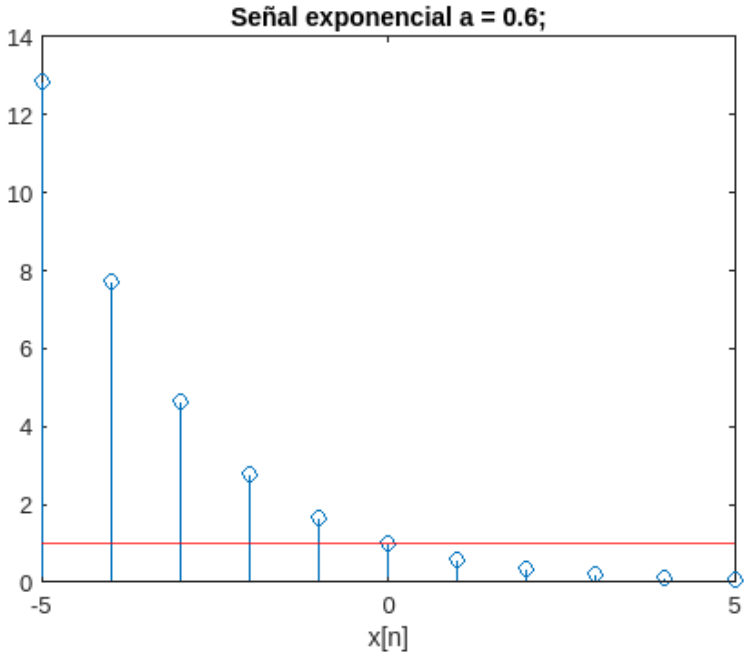
Señales exponenciales y sinusoidales discretas

Exponenciales:

```
n = -5:5;
C = 1;
a = 1.5;
y = C*a.^n;
figure
stem(n,y)
hold on
yline(1,'-r')                          % Línea horizontal en y = 1
title('Señal exponencial a = 1.5;')    % de color rojo.
xlabel('x[n]')
```

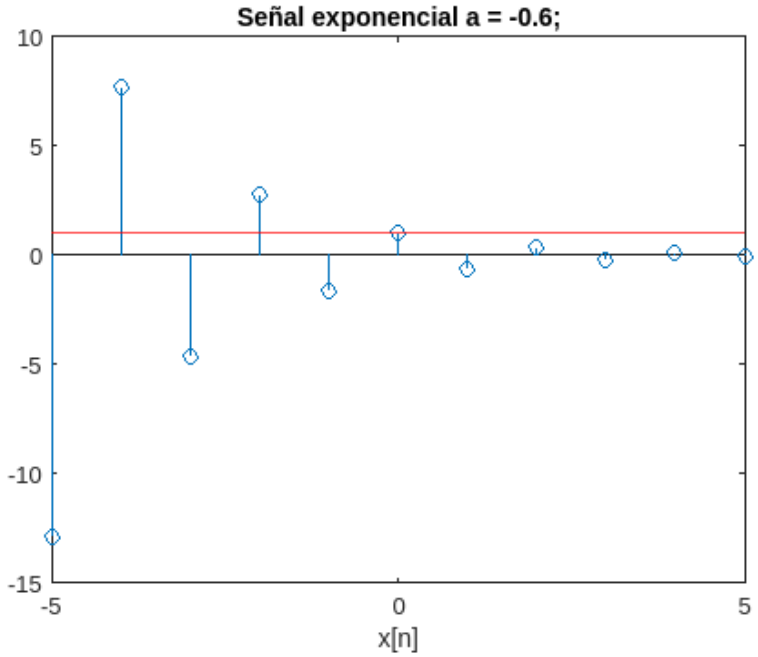



```
a = 0.6;
y = C*a.^n;
figure
stem(n,y)
hold on
yline(1,'-r')
title('Señal exponencial a = 0.6;')
xlabel('x[n]')
```

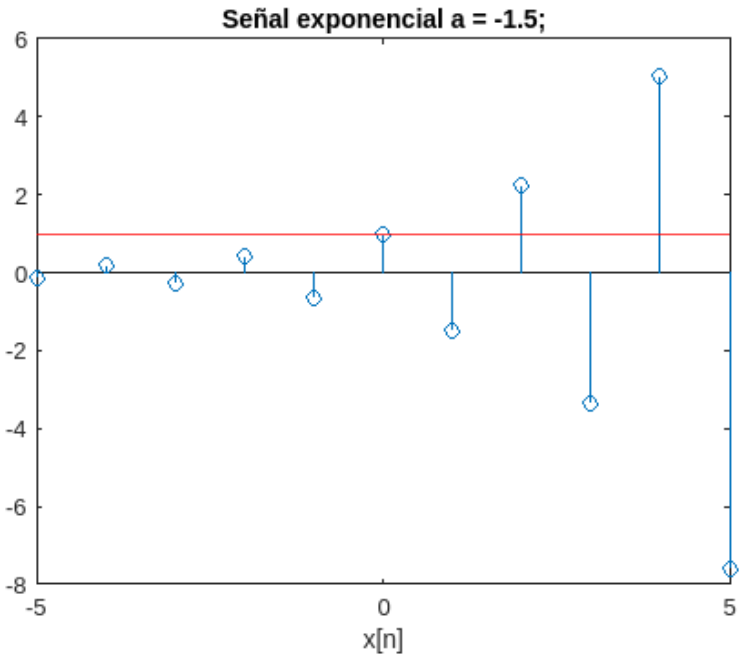



```
a = -0.6;
```

```
y = C*a.^n;
figure
stem(n,y)
hold on
yline(1,'-r')
title('Señal exponencial a = -0.6;')
xlabel('x[n]')
```

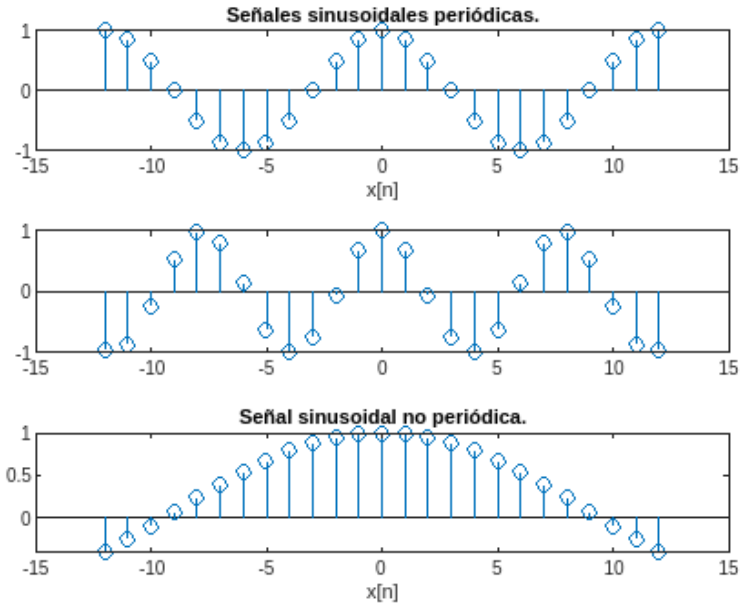



```
a = -1.5;
y = C*a.^n;
figure
stem(n,y)
hold on
yline(1,'-r')
title('Señal exponencial a = -1.5;')
xlabel('x[n]')
```



Sinusoidales:

```
n = -12:12;
y = cos(2*pi*n/12);
figure
subplot(3,1,1)
stem(n,y)
title('Señales sinusoidales periódicas.')
xlabel('x[n]')
y = cos(8*pi*n/31);
subplot(3,1,2)
stem(n,y)

y = cos(n/6);                % No periódica.
subplot(3,1,3)
stem(n,y)
title('Señal sinusoidal no periódica.')
xlabel('x[n]')
```



Más sobre MATLAB

Un aspecto importante en MATLAB es la posibilidad de realizar cálculos mediante vectorización. Esta característica hace uso de la expansión implícita donde las dimensiones no congruentes son interpretadas de todas maneras:

```
clearvars

a = [1:5;6:10]        % 2x5
```

```
a = 2×5
     1     2     3     4     5
     6     7     8     9    10
```

```
b = a.*(ones(1,5)*2)  % 2x5 * 1x5 ? -> Multiplicación por filas.
```

```
b = 2×5
     2     4     6     8    10
    12    14    16    18    20
```

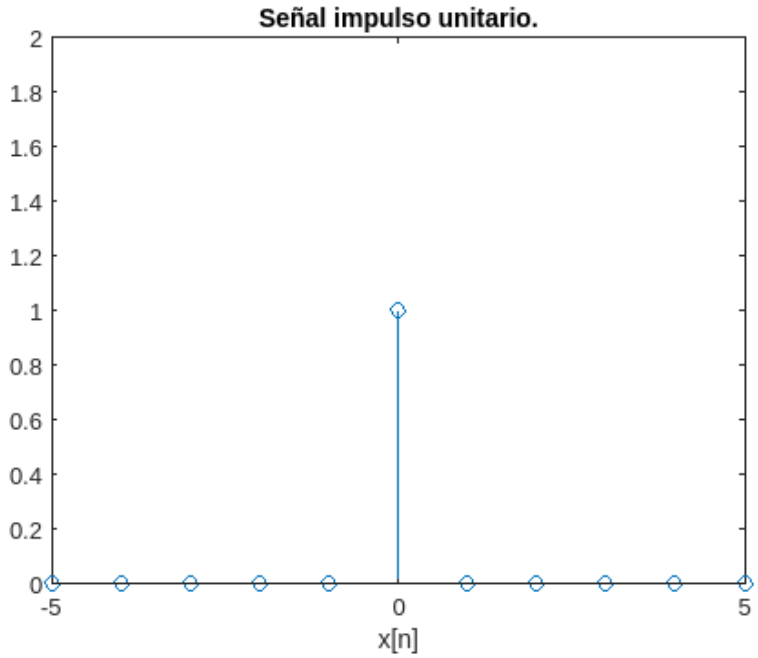
```
c = a.*([1;1]*3)      % 2x5 * 2x1 ? ->      ''     por columnas.
```

```
c = 2×5
     3     6     9    12    15
    18    21    24    27    30
```

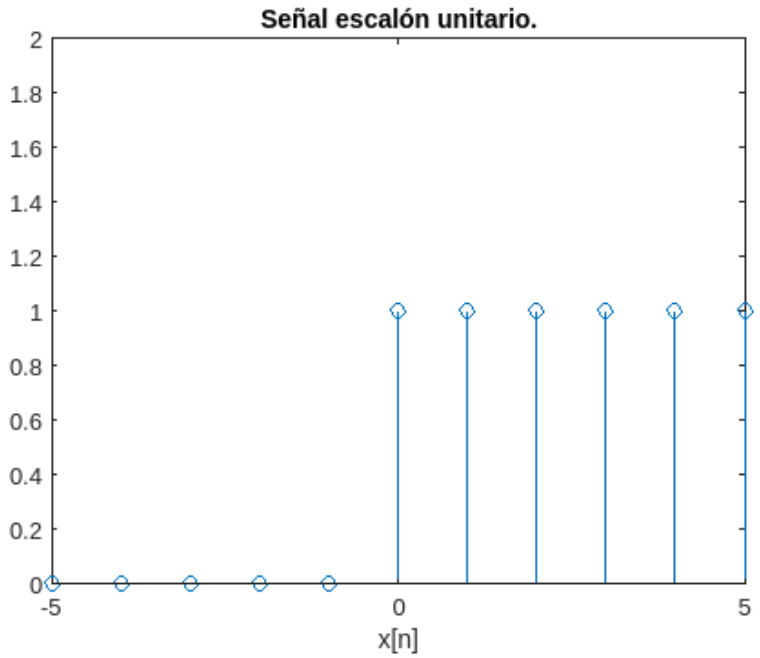
Además, podemos utilizar operadores condicionales para modificar vectores y matrices:

```
clearvars

n = -5:5;
x_n = n==0;                          % Arreglo tipo lógico 1 en n==0.
```

```
figure
stem(n,x_n)
title('Señal impulso unitario.')
xlabel('x[n]')
ylim([0 2])
```

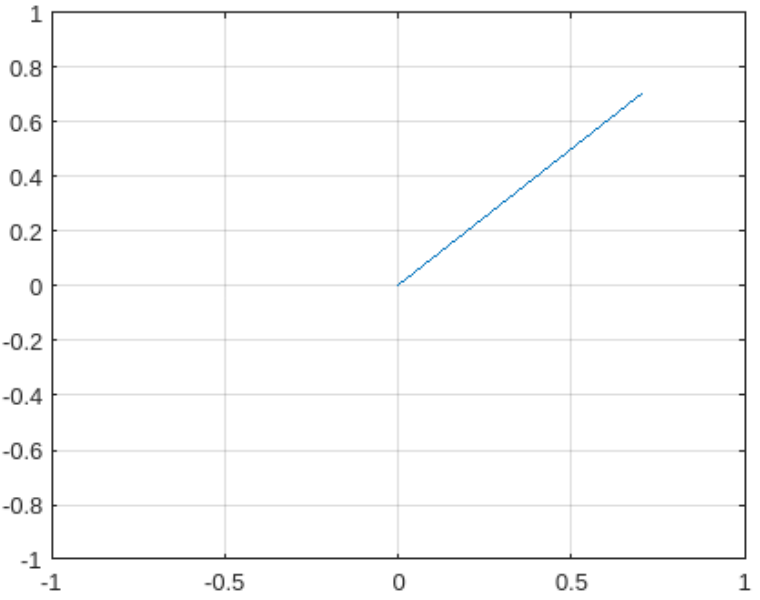



```
x_n = n>=0;                             % Arreglo tipo lógico 1 en n>=0.

figure
stem(n,x_n)
title('Señal escalón unitario.')
xlabel('x[n]')
ylim([0 2])
```



Finalmente, la representación de números complejos en Matlab se realiza mediante el operador 1i:

```
a = 0.707;
b = 0.707;
C_1 = a+1i*b;                          % Variable tipo double (complex).

figure
plot([0 real(C_1)], [0 imag(C_1)])   % Partes reales e imaginarias.
xlim([-1 1])
ylim([-1 1])
grid on
```



Referencias

[1] Oppenheim, A., Willsky, A., & Nawab, S. (1998). Signals and Systems, (2nd ed.). Prentice Hall. [Hernández, G.M. (Tr.), originalmente publicado en inglés].