3 Independencia lineal, base ortonormal, combinación lineal.

Objetivo.

Revisar e ilustrar los conceptos de independencia lineal y base ortonormal para $\mathbb{R}^2$ usando la biblioteca `numpy`.

Trabajo realizado con el apoyo del Programa UNAM-DGAPA-PAPIME PE101922

```
# Importamos las bibliotecas requeridas
import numpy as np
import ipywidgets as widgets
import macti.visual as mvis
from macti.evaluation import *
```

```
quizz = Quizz('03', 'notebooks', 'local')
```

3.1 Independencia lineal.

Los vectores $\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n$ son **linealmente independientes** si de la ecuación:

$$\sum_{i=1}^{n} \alpha_i \vec{x}_i = 0 \tag{1}$$

se deduce que $\alpha_i = 0$, para toda i. Si por lo menos una de las α_i es distinta de cero, entonces los vectores son **linealmente dependientes**.

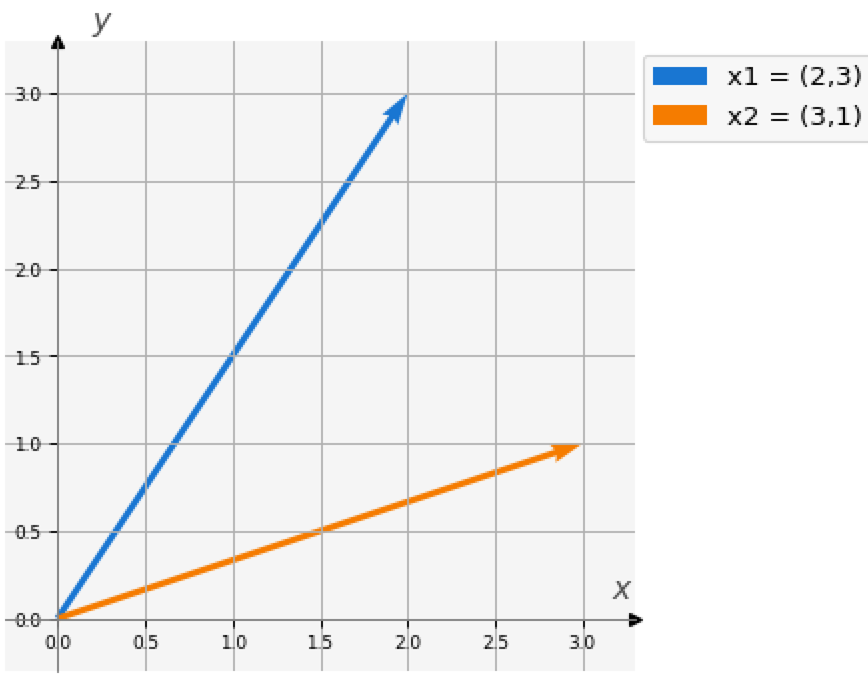
3.2 Ejemplo 1.

Definimos dos vectores, $\vec{x}_1 = (2,3)$ y $\vec{x}_2 = (3,1)$ en $\mathbb{R}^2$ usando numpy como sigue:

```
x1 = np.array([2, 3])
x2 = np.array([3, 1])

# Imprimimos los vectores
print('x1 = {}'.format(x1))
print('x2 = {}'.format(x2))
```

```
        # Visualizamos los vectores.
        v = mvis.Plotter()  # Definición de un objeto para crear figuras.
        v.set_coordsys(1)   # Definición del sistema de coordenadas.
        v.plot_vectors(1, [x1, x2], ['x1 = (2,3)', 'x2 = (3,1)'], ofx=-0.1) # Graficación
        v.grid()  # Muestra la rejilla del sistema de coordenadas.
```

```
x1 = [2 3]
x2 = [3 1]
```



Observa que los vectores **no son paralelos**, esto es equivalente a que los vectores sean **linealmente independientes**.

Ahora, de acuerdo con la definión (1) tenemos que $\alpha_1\vec{x}_1 + \alpha_2\vec{x}_2 = 0$ solo se cumple cuando $\alpha_1 = \alpha_2 = 0$. La siguiente celda de código, genera un interactivo en donde se muestra lo anterior de manera gráfica para $\alpha_1, \alpha_2 \in [-2, 2]$. Ejecuta la celda y posteriormente mueve el valor de las α's.

```
        def dependencial_lineal(x1, x2, α1, α2):
            print('x1 = {} \t x2 = {}'.format(x1, x2))
            print('α1 * x1 + α2 * x2 = {}'.format(α1 * x1 + α2 * x2))
            # Visualizamos los vectores.
            v = mvis.Plotter()  # Definición de un objeto para crear figuras.
            v.set_coordsys(1)   # Definición del sistema de coordenadas.
            v.plot_vectors_sum(1, [α1 * x1, α2 * x2], ['α1 * x1', 'α2 * x2'], ofx=-0.1) #
            v.grid()  # Muestra la rejilla del sistema de coordenadas.
            return

        widgets.interactive(dependencial_lineal,
                            x1 = widgets.fixed(x1),
                            x2 = widgets.fixed(x2),
                            α1 = widgets.FloatSlider(min=-2.0, max=2.0, step=0.5, value=1
                            α2 = widgets.FloatSlider(min=-2.0, max=2.0, step=0.5, value=1
```

3.3 Base ortonormal

En el espacio euclidiano $\mathbb{R}^n$, los vectores $\vec{e}_1 = (1, 0, \dots, 0), \vec{e}_2 = (0, 1, \dots, 0), \dots, \vec{e}_n = (0, 0, \dots, n)$, son linealmente independientes y representan una **base ortonormal**. Además, cualquier vector $\vec{z} = (z_1, z_2, \dots, z_n) \in \mathbb{R}^n$ se puede representar como

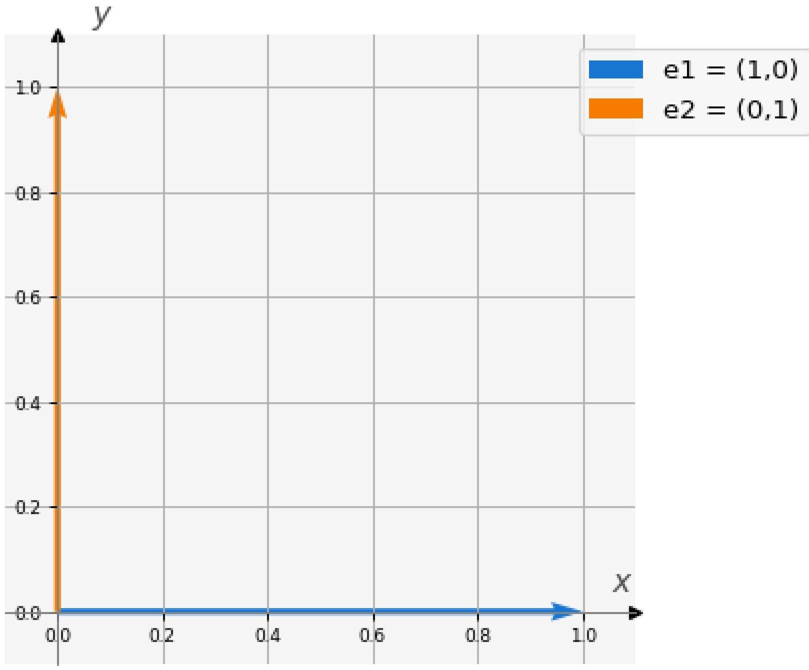
$$\vec{z} = \sum_{i=1}^{n} z_i \vec{e}_i$$

3.4 Ejemplo 2.

Definimos los vectores: $\vec{e}_1 = (1, 0)$ y $\vec{e}_1 = (1, 0)$ como sigue:

```
e1 = np.array([1, 0])
e2 = np.array([0, 1])

# Imprimimos los vectores
print('e1 = {}'.format(e1))
print('e2 = {}'.format(e2))

# Visualizamos los vectores.
v = mvis.Plotter()  # Definición de un objeto para crear figuras.
v.set_coordsys(1)   # Definición del sistema de coordenadas.
v.plot_vectors(1, [e1, e2], ['e1 = (1,0)', 'e2 = (0,1)'],ofx=-0.2) # Graficación
v.grid()  # Muestra la rejilla del sistema de coordenadas.
```

```
e1 = [1 0]
e2 = [0 1]
```



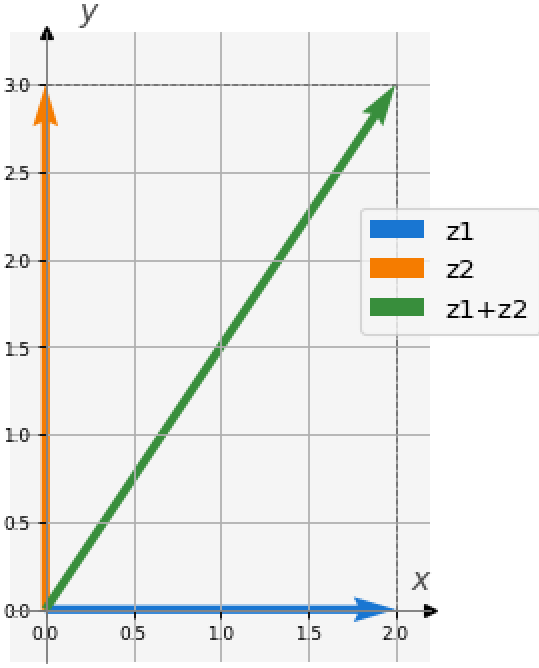
Observa que los vectores **son ortogonales** y de tamaño unitario, por lo que representan una base ortonormal de $\mathbb{R}^2$.

Con esta base podemos representar cualquier vector de $\mathbb{R}^2$, particularmente $\vec{x}_1 = (2,3)$ y $\vec{x}_2 = (3,1)$ del **Ejemplo 1**.

Construcción del vector $\vec{x}_1$:

```
# Construcción de la combinación lineal
z1 = x1[0] * e1
z2 = x1[1] * e2

# Resultado final
z = z1 + z2

# Imprimimos el vector
print('x1 = {}'.format(z))

# Visualizamos los vectores.
v = mvis.Plotter()  # Definición de un objeto para crear figuras.
v.set_coordsys(1)   # Definición del sistema de coordenadas.
v.plot_vectors_sum(1, [z1, z2], ['z1', 'z2'],ofx=-0.2, w=0.02) # Graficación de l
v.grid()  # Muestra la rejilla del sistema de coordenadas.
```

```
x1 = [2 3]
```



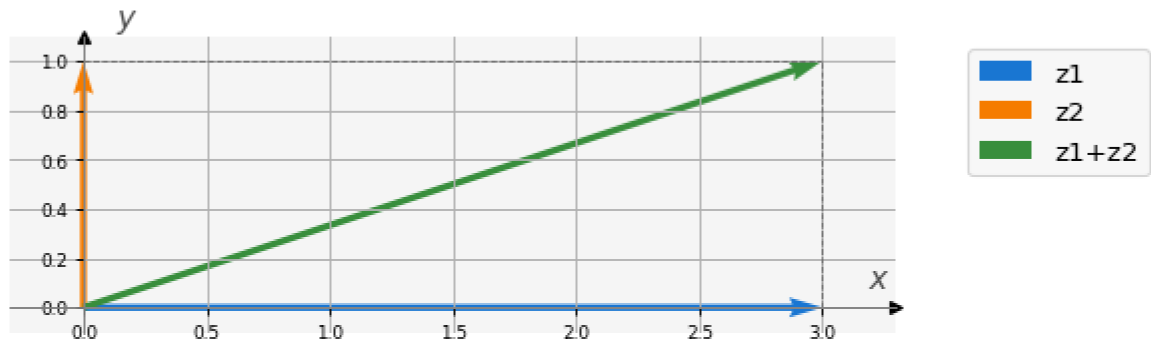
3.5 Ejercicio 1.

Construye $\vec{x}_2$ usando la base ortonormal definida en el ejemplo 2 como se hizo para $\vec{x}_1$.

```
# Construcción de la combinación lineal
# z1 = ...
# z2 = ...

# Resultado final
# z = z1 + z2

### BEGIN SOLUTION
# Construcción de la combinación lineal
z1 = x2[0] * e1
z2 = x2[1] * e2

# Resultado final
z = z1 + z2

file_answer =FileAnswer()
file_answer.write('1', z, 'z es incorrecta: revisa la construcción de la combinac
### END SOLUTION

# Imprimimos el vector
print('x2 = {}'.format(z))

# Visualizamos los vectores.
```

```
v = mvis.Plotter()  # Definición de un objeto para crear figuras.
v.set_coordsys(1)   # Definición del sistema de coordenadas.
v.plot_vectors_sum(1, [z1, z2], ['z1', 'z2'],ofx=-0.2, w=0.0075) # Graficación de
v.grid()  # Muestra la rejilla del sistema de coordenadas.
```

```
Creando el directorio :/home/jovyan/macti/notebooks/.ans/Algebra_Lineal_01/
Respuestas y retroalimentación almacenadas.
x2 = [3 1]
```




```
quizz.eval_numeric('1', z)
```

```
----------------------------------------
1 | Tu resultado es correcto.
----------------------------------------
```

3.6 Ejercicio 2.

Usando `numpy` define la base ortonormal $\{\vec{e}_1, \vec{e}_2, \vec{e}_3, \vec{e}_4\} \in \mathbb{R}^4$ y con ella construye el vector $\vec{y} = (1.5, 1.0, 2.3, -1.0)$. Imprime la base ortonormal y el resultado de construir el vector $\vec{y}$.

```
### BEGIN SOLUTION
e1 = np.array([1, 0, 0, 0])
e2 = np.array([0, 1, 0, 0])
e3 = np.array([0, 0, 1, 0])
e4 = np.array([0, 0, 0, 1])

# Imprimimos los vectores
print('e1 = {}'.format(e1))
print('e2 = {}'.format(e2))
print('e3 = {}'.format(e3))
print('e4 = {}'.format(e4))

y1 = 1.5 * e1
y2 = 1.0 * e2
```

```
        y3 = 2.3 * e3
        y4 =-1.0 * e4

        print(' y = {}'.format(y1 + y2 + y3 + y4))

        file_answer.write('2', e1, 'e1 es incorrecto: revisa la construcción del vector.'
        file_answer.write('3', e2, 'e2 es incorrecto: revisa la construcción del vector.'
        file_answer.write('4', e3, 'e3 es incorrecto: revisa la construcción del vector.'
        file_answer.write('5', e4, 'e4 es incorrecto: revisa la construcción del vector.'
        ### END SOLUTION
```

```
e1 = [1 0 0 0]
e2 = [0 1 0 0]
e3 = [0 0 1 0]
e4 = [0 0 0 1]
 y = [ 1.5  1.   2.3 -1. ]
El directorio :/home/jovyan/macti/notebooks/.ans/Algebra_Lineal_01/ ya existe
Respuestas y retroalimentación almacenadas.
```

```
        quizz.eval_numeric('2', e1)
        quizz.eval_numeric('3', e2)
        quizz.eval_numeric('4', e3)
        quizz.eval_numeric('5', e4)
```

```
----------------------------------------
2 | Tu resultado es correcto.
----------------------------------------
----------------------------------------
3 | Tu resultado es correcto.
----------------------------------------
----------------------------------------
4 | Tu resultado es correcto.
----------------------------------------
----------------------------------------
5 | Tu resultado es correcto.
----------------------------------------
```