

# Proyecto Álgebra Lineal

GitHub Repository - jgarnigar

## Descripción

- Los datos fueron encriptados multiplicando una matriz  $6 \times 6$  por una matriz  $6 \times 1$ . El resultado es una matriz  $6 \times 1$  con los datos encriptados. Estos datos nuevos ocultan una imagen. La nueva matriz  $6 \times 1$  nos da los valores  $(x, y)$ .
  - Para descryptar estos datos, obtendremos la matriz inversa de la matriz  $6 \times 6$  y multiplicaremos por la  $6 \times 1$  con los datos encriptados.
  - Finalmente usaremos **Matplotlib** con el fin de visualizar los datos descryptados.
- 

## Índice

1. Descripción
2. Instalación y requisitos
3. Objetivo
4. Estructura
5. Operaciones matemáticas
6. Métodos de Numpy
7. Datos
8. Ecuaciones
  - a) Primera ecuación
  - b) Segunda ecuación
  - c) Tercera ecuación
  - d) Cuarta ecuación
  - e) Resultado de las ecuaciones
9. Clases
  - a) Librerías
  - b) Cifrado()
  - c) Crear\_Array()

- d)* Ecuacion()
- e)* Desempaquetar\_Array()
- f)* Rotacion()
- g)* Traslacion()
- h)* GuardarDatos()
- i)* App()
- j)* Graficar()

#### 10. Instancias

#### 11. Comprobación de datos

- a)* Matriz inversa
- b)* Desencriptar datos
- c)* Rotación
- d)* Traslación

#### 12. Desencriptar datos

#### 13. Gráficas - Datos Resueltos!

- a)* Gráfico desencriptado
- b)* Gráfico Matriz Rotación
- c)* Gráfico Matriz Rotación y Traslado

#### 14. Guardar Datos

#### 15. Conclusiones

#### 16. Autor

#### 17. Anexo A: Estructura del Proyecto

#### 18. Anexo B: Tecnologías Empleadas

#### 19. Anexo C: Fundamentos Teóricos

## Instalación y requisitos

Utilizaremos **Python 3.10** o superior, **Numpy** y **Matplotlib**. Para asegurarnos que todo funcione, por favor cree un entorno virtual.

### Instalar Python 3.10 o superior

Descargar Python

### Clone el repositorio:

```
1 git clone https://github.com/jgarnigar/Proyecto-Algebra-Lineal.git
```

## Crear el entorno virtual

```
1 python -m venv venv
2 source venv/bin/activate      # Para Linux/Mac
3 venv\Scripts\activate         # Para Windows
```

## Instale los requerimientos

```
1 pip install -r requirements.txt
```

## Ejecución del programa

Una vez tengamos el repositorio clonado y todos los requisitos instalados, ejecute este código desde la carpeta principal:

```
1 python main.py
```

**Nota:** asegúrese de ejecutar este comando desde la carpeta raíz del proyecto.

---

## Objetivo

- El objetivo es crear un algoritmo a través de **Python** y **Numpy** para poder descryptar estos datos de forma automatizada y por último mostrar estos datos con **Matplotlib**.
- 

## Estructura

### Proyecto Álgebra Lineal

datos/	→ Datos encriptados y descryptados
funciones/	→ Funciones lógicas
main.py	→ Archivo principal de ejecución
requirements.txt	
README.md	

---

## Operaciones matemáticas

Estos son los temas utilizados para resolver el proyecto.

- Multiplicación de matrices.
  - Matrices inversas.
  - Determinantes.
  - Sistema de ecuaciones.
  - Producto escalar.
  - Producto vectorial.
-

# Métodos de Numpy.

Explicaremos brevemente los métodos utilizados en **Numpy** para poder trabajar con nuestras clases.

- `np.linalg.inv()`  $\Rightarrow$  Inversa de una matriz.
- `matriz_a @ matriz_b`  $\Rightarrow$  Multiplicación de matrices.
- `np.array()`  $\Rightarrow$  Creación de arrays.
- `np.linalg.solve(primer_termino, segundo_termino)`  $\Rightarrow$  Resolución de sistema de ecuaciones.
- `np.ravel()`  $\Rightarrow$  Convierte un array multidimensional en uno unidimensional.
- `np.deg2rad()`  $\Rightarrow$  Convierte un ángulo en grados a radianes.
- `np.cos()`  $\Rightarrow$  Función Coseno.
- `np.sen()`  $\Rightarrow$  Función Seno.
- `np.reshape()`  $\Rightarrow$  Función para transformar la dimensión de arrays. Ejemplo, pasar de  $1 \times 6$  a  $6 \times 1$ .
  - Si el array es un  $1 \times 6$ , usamos `np.reshape(-1, 1)`:  $-1$  significa que toma el máximo valor que en este caso es 6 que serán las filas, y 1, que será el número total de columnas. Ahora el array es de  $6 \times 1$ .

---

## Datos

*Todos los datos fueron almacenados dentro de la carpeta datos.*

```
Proyecto-algebra-linea
datos > datos encriptados.txt
valores descriptados.txt
valores rotados.txt
valores trasladados.txt
```

---

## Ecuaciones

Antes de empezar a descryptar los datos, tenemos que tener todos los datos de nuestra matriz  $6 \times 6$  la cual usaremos para resolver el proyecto.

Tenemos la siguiente matriz la cual tiene incógnitas que hay que encontrar:

$$A = \begin{pmatrix} a & 3 & b & 5 & c & 8 \\ 6 & d & 0 & e & 7 & f \\ g & 8 & h & 1 & i & 7 \\ 11 & j & 8 & k & 12 & m \\ n & -1 & p & -5 & r & 3 \\ 4 & t & 2 & w & 9 & z \end{pmatrix}$$

## Primera Ecuación

Resolvemos nuestra ecuación de  $10 \times 10$

$$\begin{aligned}6a + 7b - c - 12d + 14e + 5f - 12g - 3h + 9i - 5j &= 48 \\2a - 15b + 8c + 6d - 7e + 9f - 9g + 5h - 8i - 6j &= 64 \\-25a + 10b - 9d - 12e + 14f - 6g + 8h - 13i + 4j &= -132 \\6a - 3b + 5c - 16d + e + 9f - 7g + 3h - 4i + 5j &= -75 \\8a - 9b + 6c - d - e - 5f + 7g + 3i + 2j &= -16 \\-5a + 6b + 9c - 2d + 10e - 14f + 3g + 5h - 12i + 6j &= -408 \\-4a + 5b + 8c - 2d + 9e - 8f + 4g + h - 2j &= -203 \\a + b + 2c - 3d + 4e - f - 4g - 7h + 2i - 4j &= 59 \\10a + 5b - 9c + 6d + e + f + 7g - 8h + 3i + 11j &= 126 \\-2a + 4b + 3c + 5d - 10e - f + 3g - h - 7i + j &= 2\end{aligned}$$

Código para resolver:

```
1 # Pasamos todos los datos del primer termino a un array.
2 primer_termino = np.array([
3     [6, 7, -1, -12, 14, 5, -12, -3, 9, -5],
4     [2, -15, 8, 6, -7, 9, -9, 5, -8, -6],
5     [-25, 10, 0, -9, -12, 14, -6, 8, -13, 4],
6     [6, -3, 5, -16, 1, 9, -7, 3, -4, 5],
7     [8, -9, 6, -1, -1, -5, 7, 0, 3, 2],
8     [-5, 6, 9, -2, 10, -14, 3, 5, -12, 6],
9     [-4, 5, 8, -2, 9, -8, 4, 1, 0, -2],
10    [1, 1, 2, -3, 4, -1, -4, -7, 2, -4],
11    [10, 5, -9, 6, 1, 1, 7, -8, 3, 11],
12    [-2, 4, 3, 5, -10, -1, 3, -1, -7, 1]
13 ], dtype=int)
14
15 # Ahora pasamos los datos del segundo termino a otro array.
16 segundo_termino = np.array([48, 64, -132, -75, -16, -408, -203, 59, 126, 2],
17                             dtype=int)
18
19 # Tenemos una clase Ecuacion() la cual hace uso de np.linalg.solve para
20 # resolver el sistema.
21 ecuacion = Ecuacion()
22
23 # la matriz de los resultados tales que.
24 # (2, 2, 3, 5, 9, 4, 7, 10, 11, 0)
```

**Nota:** La clase Ecuacion() se verá más adelante.

**Incógnitas adicionales:** Para las incógnitas  $k, m, n, p, r, t, w, z$  debe resolver las siguientes operaciones entre vectores:

## Segunda Ecuación

Se tienen los vectores  $\vec{U} = (3, 6, 7)$  y  $\vec{V} = (k, m, n)$ . El resultado de operar  $2\vec{U} \times 3\vec{V}$  es igual a  $612\hat{i} + 156\hat{j} - 396\hat{k}$  y el producto  $\vec{U} \cdot \vec{V} = 58$ .

$$2\vec{U} \times 3\vec{V} = (612, 156, -396)$$

Dividiendo por 6:

$$\frac{2\vec{U} \times 3\vec{V}}{6} = (102, 26, -66)$$

Sistema de ecuaciones del producto cruz:

$$\begin{aligned}6n - 7m &= 102 \\7k - 3n &= 26 \\3m - 6k &= -66\end{aligned}$$

Despejando  $k$  y  $m$  en términos de  $n$ :

$$m = \frac{6n - 102}{7} \quad \text{y} \quad k = \frac{3n + 26}{7}$$

Usando el producto escalar  $\vec{U} \cdot \vec{V} = 58$ :

$$3k + 6m + 7n = 58$$

Sustituyendo  $k$  y  $m$ :

$$3\left(\frac{3n + 26}{7}\right) + 6\left(\frac{6n - 102}{7}\right) + 7n = 58$$

Multiplicando por 7:

$$\begin{aligned}3(3n + 26) + 6(6n - 102) + 49n &= 406 \\9n + 78 + 36n - 612 + 49n &= 406 \\94n &= 406 - 78 + 612 \Rightarrow 94n = 940 \Rightarrow n = 10\end{aligned}$$

Con  $n = 10$ :

$$\begin{aligned}m &= \frac{6(10) - 102}{7} = \frac{60 - 102}{7} = \frac{-42}{7} \Rightarrow m = -6 \\k &= \frac{3(10) + 26}{7} = \frac{30 + 26}{7} = \frac{56}{7} \Rightarrow k = 8\end{aligned}$$

**Resultado:**

$$k = 8, \quad m = -6, \quad n = 10$$

### Tercera Ecuación

Se tienen los vectores  $\vec{U} = (6, p, r)$  y  $\vec{V} = (t, 8, 9)$ . El resultado de operar  $3\vec{U} - 10\vec{V}$  es igual a  $-42\hat{i} - 68\hat{j} - 126\hat{k}$ .

$$3\vec{U} - 10\vec{V} = (3(6) - 10t, 3p - 10(8), 3r - 10(9)) = (-42, -68, -126)$$

**Componente  $i$  ( $t$ ):**

$$18 - 10t = -42 \Rightarrow -10t = -60 \Rightarrow t = 6$$

**Componente  $j$  ( $p$ ):**

$$3p - 80 = -68 \Rightarrow 3p = 12 \Rightarrow p = 4$$

**Componente  $k$  ( $r$ ):**

$$3r - 90 = -126 \Rightarrow 3r = -36 \Rightarrow r = -12$$

**Resultado:**

$$t = 6, \quad p = 4, \quad r = -12$$

## Cuarta Ecuación

Se tienen los vectores:

$$\vec{U} = \left( \frac{-1}{2}, \frac{\sqrt{38}}{2}, \frac{5}{2} \right) \quad \text{y} \quad \vec{V} = (11, \sqrt{342}, -21)$$

La magnitud de  $|\vec{U}|$  es igual a  $-w$ . Además, el producto escalar  $\vec{U} \cdot \vec{V}$  es igual a  $z$ . **Cálculo de  $w$  (magnitud):**

$$|\vec{U}| = \sqrt{\left(\frac{-1}{2}\right)^2 + \left(\frac{\sqrt{38}}{2}\right)^2 + \left(\frac{5}{2}\right)^2} = \sqrt{\frac{1}{4} + \frac{38}{4} + \frac{25}{4}} = \sqrt{\frac{64}{4}} = \sqrt{16} = 4$$

Dado que  $|\vec{U}| = -w$ :

$$4 = -w \Rightarrow w = -4$$

**Cálculo de  $z$  (producto escalar):**

$$\vec{U} \cdot \vec{V} = z$$

Notemos que  $\sqrt{342} = \sqrt{9 \times 38} = 3\sqrt{38}$ .

$$\begin{aligned} z &= \left(\frac{-1}{2}\right)(11) + \left(\frac{\sqrt{38}}{2}\right)(3\sqrt{38}) + \left(\frac{5}{2}\right)(-21) \\ z &= \frac{-11}{2} + \frac{3(\sqrt{38})^2}{2} + \frac{-105}{2} = \frac{-11}{2} + \frac{3(38)}{2} + \frac{-105}{2} \\ z &= \frac{-11 + 114 - 105}{2} = \frac{-2}{2} = -1 \end{aligned}$$

**Resultado:**

$$w = -4, \quad z = -1$$

## Resultado de las ecuaciones.

**Variables Encontradas:**

$$a = 2, \quad b = -2, \quad c = -3, \quad d = 5, \quad e = -9, \quad f = 4, \quad g = -7, \quad h = -10, \quad i = 11, \quad j = 0, \quad k = 8,$$

**Matriz para codificar los vectores:**

$$\begin{pmatrix} 2 & 3 & -2 & 5 & -3 & 8 \\ 6 & 5 & 0 & -9 & 7 & 4 \\ -7 & 8 & -10 & 1 & 11 & 7 \\ 11 & 0 & 8 & 8 & 12 & -6 \\ 10 & -1 & 4 & -5 & -12 & 3 \\ 4 & 6 & 2 & -4 & 9 & -1 \end{pmatrix}$$

---

## Clases

*Ahora que ya resolvimos nuestras ecuaciones y obtuvimos nuestra matriz completa, ya podemos comenzar a programar para poder descryptar nuestros datos.*



## Librerías

Primero importamos las librerías que usaremos a lo largo de nuestro proyecto.

```
1 import numpy as np
2 import matplotlib.pyplot as plt
```

## Cifrado()

Creamos nuestra clase **Cifrado** la cual cifra, descifra datos y podemos obtener una matriz inversa para depurar.

### Funcionamiento de la clase Cifrado()

- La función `inversa()` devuelve la inversa de la matriz ingresada (usando `np.linalg.inv()`).
- La función `cifrar()` obtiene 2 matrices y las multiplica (usando `@`) devolviendo la matriz cifrada.
- La función `descifrar()` calcula la inversa de la matriz de codificación y la multiplica por la matriz cifrada para devolver la matriz descifrada.

```
1 class Cifrado():
2
3     def inversa(self, matriz_codificar):
4         self.matriz_codificar = matriz_codificar
5         matriz_inversa = np.linalg.inv(self.matriz_codificar)
6         return matriz_inversa
7
8     def cifrar(self, matriz, matriz_codificar):
9         self.matriz = matriz
10        self.matriz_codificar = matriz_codificar
11        matriz_cifrada = self.matriz_codificar @ self.matriz
12        return matriz_cifrada
13
14    def descifrar(self, matriz_codificacion, matriz_resolver):
15        self.matriz_codificacion = matriz_codificacion
16        self.matriz_resolver = matriz_resolver
17        matriz_inversa = np.linalg.inv(self.matriz_codificacion)
18        matriz_descifrada = matriz_inversa @ self.matriz_resolver
19        return matriz_descifrada
```

## Crear\_Array()

La clase `crear_array()` fue creada con el objetivo de obtener un arreglo con valores alternados de  $(x, y)$  en un formato  $(x_1, y_1, x_2, y_2 \dots)$  y los separará en dos arreglos independientes con las coordenadas  $(x)$  y  $(y)$ .

### Funcionamiento de la clase Crear\_Array()

- Método `valores_x()`: Recorre el arreglo original y toma los valores impares (índice 1, 3, 5...).
- Método `valores_y()`: Recorre el arreglo original y toma los valores pares (índice 2, 4, 6...).



```

1 class crear_array():
2
3     def valores_x(self, array):
4         self.array = array
5         puntos_x = []
6         contador = 1
7         for x in self.array:
8             if contador % 2 != 0:
9                 puntos_x.append(x)
10                contador += 1
11        return puntos_x
12
13     def valores_y(self, array):
14         self.array = array
15         puntos_y = []
16         contador = 1
17         for y in self.array:
18             if contador % 2 == 0:
19                 puntos_y.append(y)
20                contador += 1
21        return puntos_y

```

## Ecuacion()

*Creamos una clase llamada **Ecuacion** con la finalidad de resolver la ecuación  $10 \times 10$  brindada en las instrucciones del proyecto.*

### Funcionamiento de la clase Ecuacion()

- Recibe el primer y segundo término separados.
- Usa `np.linalg.solve()` para encontrar los valores de las incógnitas.

```

1 class Ecuacion():
2     def resolver(self, primer_termino, segundo_termino):
3         self.primer_termino = np.array(primer_termino)
4         self.segundo_termino = np.array(segundo_termino)
5         solucion = np.linalg.solve(self.primer_termino, self.segundo_termino
6     )
7     return solucion

```

## Desempaquetar\_Array()

*Esta clase transforma los arreglos de coordenadas anidados en listas unidimensionales.*

```

1 class Desempaquetar_Array():
2     def desempaquetar(self, valores_x, valores_y):
3         self.valores_x = valores_x
4         self.valores_y = valores_y
5         new_valores_x = []
6         new_valores_y = []
7
8         for nueva_lista in self.valores_x:
9             for x in np.ravel(nueva_lista):
10                new_valores_x.append(x)
11
12        for nueva_lista in self.valores_y:
13            for y in np.ravel(nueva_lista):

```

```

14         new_values_y.append(y)
15
16     return new_values_x, new_values_y

```

## Rotacion()

La clase *Rotacion* aplica una transformación geométrica de rotación a un conjunto de puntos en el plano cartesiano.

### Matriz de Rotación

La matriz de rotación en 2D está definida como:

$$R(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

Y la transformación de un punto  $(x, y)$  se obtiene mediante:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = R(\theta) \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

### Funcionamiento de la clase Rotacion()

```

1 class Rotacion():
2     def rotar_matriz(self, valores_x, valores_y):
3         self.valores_x = valores_x
4         self.valores_y = valores_y
5         angulo = 135
6         angulo_rad = np.deg2rad(angulo)
7
8         rotacion_x = []
9         rotacion_y = []
10
11        matriz_rotacion = np.array([
12            [np.cos(angulo_rad), -np.sin(angulo_rad)],
13            [np.sin(angulo_rad), np.cos(angulo_rad)]
14        ])
15
16        for x, y in zip(self.valores_x, self.valores_y):
17            new_array = np.array([x, y]).reshape(2,1)
18            array_rotado = matriz_rotacion @ new_array
19            rotacion_x.append(array_rotado[0,0])
20            rotacion_y.append(array_rotado[1,0])
21
22        return rotacion_x, rotacion_y

```

## Traslacion()

La clase *Traslacion* aplica una transformación geométrica de traslación a un conjunto de puntos en el plano cartesiano.

### Matriz de Traslación (Coordenadas Homogéneas)

$$T(a, b) = \begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = T(a, b) \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

### Funcionamiento de la clase Traslacion()

```

1 class Traslacion():
2     def trasladar_matriz(self, x, y, a, b):
3         self.x = x
4         self.y = y
5         # a = movimiento en x
6         self.a = a
7         # b = movimiento en y
8         self.b = b
9
10        valores_x = []
11        valores_y = []
12
13        matriz_traslacion = np.array([
14            [1, 0, self.a],
15            [0, 1, self.b],
16            [0, 0, 1]
17        ])
18
19        for punto_x, punto_y in zip(self.x, self.y):
20            punto = np.array([punto_x, punto_y, 1]).reshape(3,1)
21            resultado = matriz_traslacion @ punto
22            valores_x.append(resultado[0,0])
23            valores_y.append(resultado[1,0])
24
25        return valores_x, valores_y

```

### GuardarDatos()

*Esta clase permite almacenar las coordenadas generadas en un archivo de texto externo en formato de pares ordenados.*

### Funcionamiento de la clase GuardarDatos()

```

1 class GuardarDatos():
2     def save(self, valores_x, valores_y, nombre_archivo):
3         self.valores_x = valores_x
4         self.valores_y = valores_y
5         self.nombre_archivo = nombre_archivo
6
7         with open(self.nombre_archivo, "w") as f:
8             for x, y in zip(valores_x, valores_y):
9                 f.write(f"({x}, {y})\n")

```

### App()

*La clase **App** se encarga de abrir un archivo de datos, convertirlos en una matriz columna y aplicar el método de cifrado o descifrado.*

## Funcionamiento de la clase App()

```
1 class App():
2     def abrir_documento(self, matriz, archivo, condicional):
3         self.archivo = archivo
4         self.matriz = matriz
5         self.condicional = condicional
6
7         array = []
8         array_valores_x = []
9         array_valores_y = []
10        resultado = []
11        matriz_original = np.array(self.matriz)
12
13        with open(self.archivo, "r") as f:
14            for line in f:
15                valores_Array = [float(x) for x in line.strip().split()]
16                array = np.array(valores_Array).reshape(-1, 1)
17
18                cir = Cifrado()
19                create = crear_array()
20
21                if self.condicional == "cifrar":
22                    resultado = cir.cifrar(array, matriz_original)
23                elif self.condicional == "descifrar":
24                    resultado = cir.descifrar(matriz_original, array)
25
26                valores_x = create.valores_x(resultado)
27                valores_y = create.valores_y(resultado)
28
29                array_valores_x.append(valores_x)
30                array_valores_y.append(valores_y)
31
32        return array_valores_x, array_valores_y
```

## Graficar()

*Esta clase nos permite graficar todos los puntos (x,y) con los datos Desencriptados, Rotados y Traslados.*

## Funcionamiento de la clase Graficar()

```
1 class Graficar():
2     def graficadora(self, x, y):
3         self.x = x
4         self.y = y
5
6         plt.scatter(self.x, self.y, color='blue', marker='o', label='Puntos
7         (x, y)')
8         # Personalizar el gr fico
9         plt.title("Gr fico de puntos")
10        plt.xlabel("Eje X")
11        plt.ylabel("Eje Y")
12        plt.legend()
13        plt.grid(True)
14
15        # Mostrar el gr fico
16        plt.show()
```



## Instancias

*Inicializamos nuestras instancias de clases:*

```
1 aplicacion = App()
2 rotar = Rotacion()
3 desempaquetador = Desempaquetar_Array()
4 graficar = Graficar()
5 traslacion = Traslacion()
6 guardar_datos = GuardarDatos()
7 cir = Cifrado()
8 create = crear_array()
```

## Comprobación de Datos

### Matriz Inversa

Buscamos comprobar que la clase Cifrado con el método `inversa` funciona para poder obtener la inversa de nuestra matriz.

**Input:**

```
1 inversa = cir.inversa(matriz_codificacion)
2 print(f"La matriz inversa es: \n{inversa}")
```

**Output:**

```
1 La matriz inversa es:
2 [[-0.13562749 -0.03955583  0.16534674  0.097423    0.19435366 -0.08729311]
3  [ 0.02119736 -0.18428661  0.04575739 -0.04399749  0.08185104  0.26227216]
4  [ 0.28413995  0.12090928 -0.33697884 -0.11187932 -0.30448355  0.15573015]
5  [ 0.03036874 -0.074251    0.03071444  0.03456945  0.01842657  0.00880997]
6  [ 0.01988791  0.10211607 -0.03490467  0.02074167 -0.08727739 -0.06304735]
7  [ 0.21047035  0.19392416 -0.17502619 -0.04965431 -0.19964907 -0.06674523]]
```

### Desencriptar datos

Multiplicaremos nuestra matriz inversa por la matriz con los datos codificados y de esta manera poder descifrarlos. La matriz con los datos encriptados es:

```
1 [230.3]
2 [263.5]
3 [238.8]
4 [814.8]
5 [-100]
6 [432.7]
```

**Input:**

```
1 #nuestra matriz de 6x1 y nuestra matriz de 6x6, usamos reshape para cambiar
  su dimensi n a 6x1
2 dato_encriptado = np.array([230.3, 263.5, 238.8, 814.8, -100, 432.7]).
  reshape(-1, 1)
3 #guardamos los datos usando la clase decifrar.
4 datos_desencriptados = cir.descifrar(matriz_codificacion, dato_encriptado)
5 #mostramos los datos por pantalla
6 print(datos_desencriptados)
```

### Output:

```
1 [[20. ]
2  [36.7]
3  [23.5]
4  [24.9]
5  [21.5]
6  [ 8.4]]
```

## Rotación de datos

A continuación, comprobaremos la **rotación de los datos** utilizando la clase `Rotacion()`.

La clase `Rotacion()` requiere `datos_x` y `datos_y`. Para obtener estos valores a partir de `datos_desencriptados`, usamos la clase `crear_array()`.

### Input: Obtención de $x$ e $y$

```
1 #Primero obtenemos los resultados (x, y), as se los podemos pasar a
   nuestra clase.
2 valor_prueba_x = create.valores_x(datos_desencriptados)
3 print(f"Los valores de x son : {valor_prueba_x}")
4 valor_prueba_y = create.valores_y(datos_desencriptados)
5 print(f"Los valores de y son: {valor_prueba_y}")
6 #estos datos a n no est n rotados, la rotaci n ser ahora con la clase
   Rotacion()
```

### Output:

```
1 Los valores de x son : [array([20.]), array([23.5]), array([21.5])]
2 Los valores de y son: [array([36.7]), array([24.9]), array([8.4])]
```

**Nota:** Podemos notar que los valores de  $(x, y)$  están en arrays anidados. Es necesario desempaquetarlos a través de la clase `Desempaquetar_Array()` antes de la rotación.

### Input: Desempaquetado y Rotación

```
1 valor_prueba_desempaquetar_x, valor_prueba_desempaquetar_y = desempaquetador
   .desempaquetar(valor_prueba_x, valor_prueba_y)
2 print(f"Los valores de x desempaquetados: {valor_prueba_desempaquetar_x}")
3 print(f"Los valores de y desempaquetados: {valor_prueba_desempaquetar_y}")
4
5 #Rotaci n de datos
6 valor_prueba_rotados_x, valor_prueba_rotados_y = rotar.rotar_matriz(
   valor_prueba_desempaquetar_x, valor_prueba_desempaquetar_y)
7 print(f"Los valores x rotados son: {valor_prueba_rotados_x}")
8 print(f"Los valores y rotados son: {valor_prueba_rotados_y}")
```

### Output:

```
1 Los valores de x desempaquetados: [np.float64(20.000000000000007), np.
   float64(23.499999999999986), np.float64(21.500000000000001)]
2 Los valores de y desempaquetados: [np.float64(36.699999999999996), np.float64
   (24.900000000000002), np.float64(8.4000000000000023)]
3 Los valores x rotados son: [np.float64(-40.09295449327722), np.float64
   (-34.22396820942889), np.float64(-21.142492757477793)]
4 Los valores y rotados son: [np.float64(-11.808683245815308), np.float64
   (-0.9899494936611762), np.float64(9.263098833543765)]
```

Con los valores rotados exitosamente, pasaremos a comprobar la **traslación**.

---

## Traslación

Ahora comprobaremos que la clase `Traslacion()` funciona pasándole los datos anteriormente rotados.

La clase `Traslacion()` recibe los datos de  $(x, y)$  y  $(a, b)$ . Donde  $(a, b)$  son las distancias de traslación para los ejes  $(x, y)$ .

### Input:

```
1 #usaremos los datos anteriormente rotados para obtener los datos finales.
2 valor_prueba_trasladado_x, valor_prueba_trasladado_y = traslacion.
   trasladar_matriz(valor_prueba_rotados_x, valor_prueba_rotados_y, 20, 30)
3
4 #pasamos los valores (a,b) como (20,30)
5 #mostramos los datos.
6
7 print(f"Los valores trasladados para x son: {valor_prueba_trasladado_x}")
8 print(f"Los valores trasladados para y son: {valor_prueba_trasladado_y}")
```

### Output:

```
1 Los valores trasladados para x son: [np.float64(-20.09295449327722), np.
   float64(-14.22396820942889), np.float64(-1.1424927574777932)]
2 Los valores trasladados para y son: [np.float64(18.19131675418469), np.
   float64(29.010050506338825), np.float64(39.263098833543765)]
```

Finalizada la traslación, procedemos a utilizar el método de automatización para **des-  
encriptar el archivo** `datos encriptados.txt`. Mostraremos las gráficas para corroborar los resultados.

---

## Gráficas de Comprobación

### Datos Desencriptados (Test)

---

```
1 #Grafica para los puntos desencriptados nicamente .
2 graficar.graficadora(valor_prueba_x, valor_prueba_y)
```

### Datos Rotados (Test)

```
1 #Grafica para los datos rotados
2 graficar.graficadora(valor_prueba_rotados_x, valor_prueba_rotados_y)
```

### Datos Traslados (Test)

```
1 #Gr fica para los datos rotados y trasladados
2 graficar.graficadora(valor_prueba_trasladado_x, valor_prueba_trasladado_y)
```



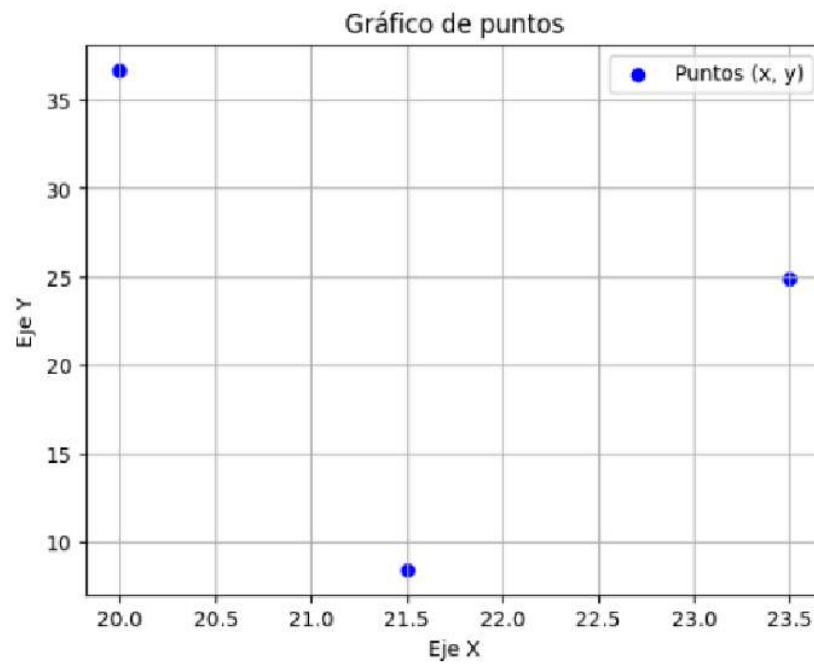


Figura 1: Gráfica de Puntos Descryptados (Test)

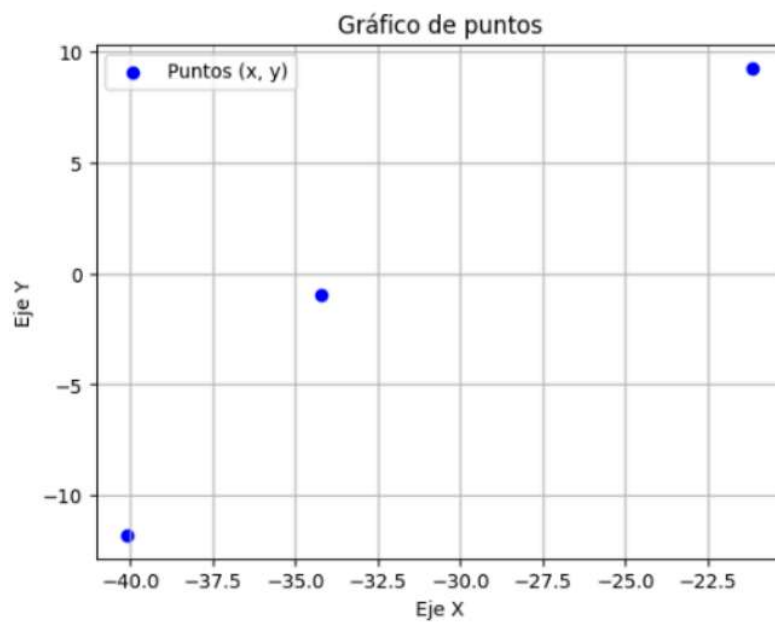


Figura 2: Gráfica de Puntos Rotados (Test)

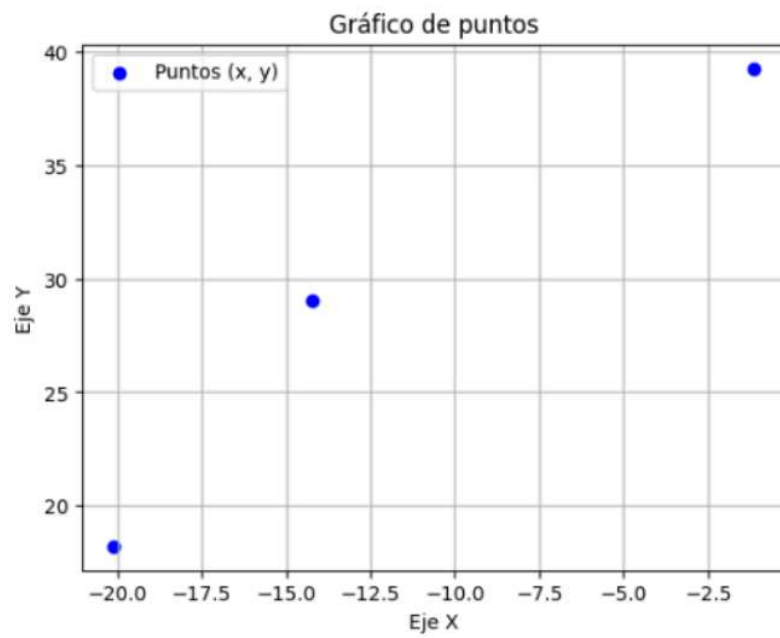


Figura 3: Gráfica de Puntos Rotados y Traslados (Test)

---

## Desencriptar Archivo

El siguiente paso es procesar el archivo `datos_encriptados.txt`, el cual contiene los datos encriptados. Cada línea tiene 6 datos que se resuelven línea por línea usando matrices  $6 \times 1$ .

### Obtención y Transformación de Datos

```
1 #Desencriptamos todos los valores para (x, y)
2 valores_x, valores_y = aplicacion.abrir_documento(matriz_codificacion, "/
   content/drive/MyDrive/Colab Notebooks/datos_encriptados.txt", "decifrar")
3
4 #Los datos estan anidados, así que los aplanamos con la clase
   Desempaquetar_Array()
5 desempaquetar_x, desempaquetar_y = desempaquetador.desempaquetar(valores_x,
   valores_y)
6
7 #Rotamos los datos ahora que estan desempaquetados
8 valores_rotados_x, valores_rotados_y = rotar.rotar_matriz(desempaquetar_x,
   desempaquetar_y)
9
10 #Trasladamos los datos ya rotados para obtener la última gráfica.
11 valores_trasladados_x, valores_trasladados_y = traslacion.trasladar_matriz(
   valores_rotados_x, valores_rotados_y, 20, 30)
```

Con todos los valores Desencriptados, Rotados y Traslados, utilizamos la clase `Graficar()` para visualizar los datos.

---

## Gráficas - Datos Resueltos

---

### Gráfico Desencriptado

```
1 graficar.graficadora(desempaquetar_x, desempaquetar_y)
```

### Gráfico Matriz Rotación

```
1 graficar.graficadora(valores_rotados_x, valores_rotados_y)
```

### Gráfico Matriz Rotación y Traslado

```
1 graficar.graficadora(valores_trasladados_x, valores_trasladados_y)
```



Figura 4: Gráfico de los Datos Descriptados Finales

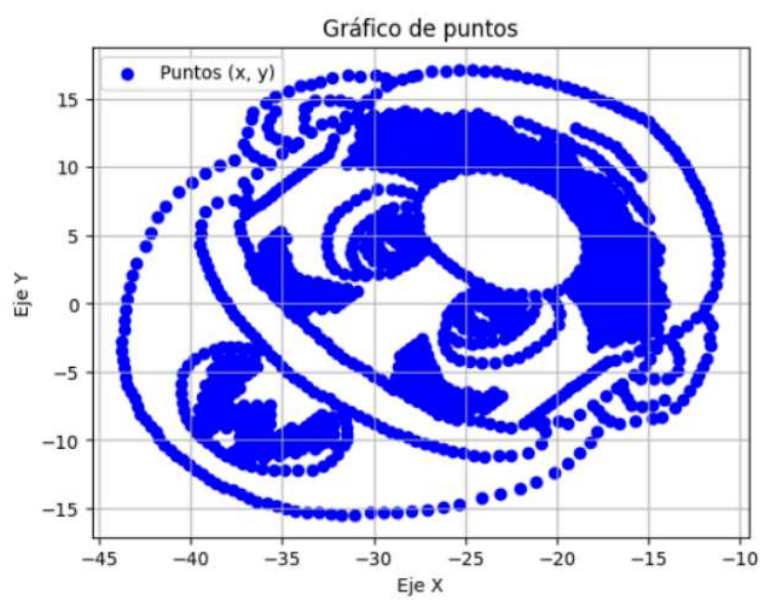


Figura 5: Gráfico de los Datos Rotados

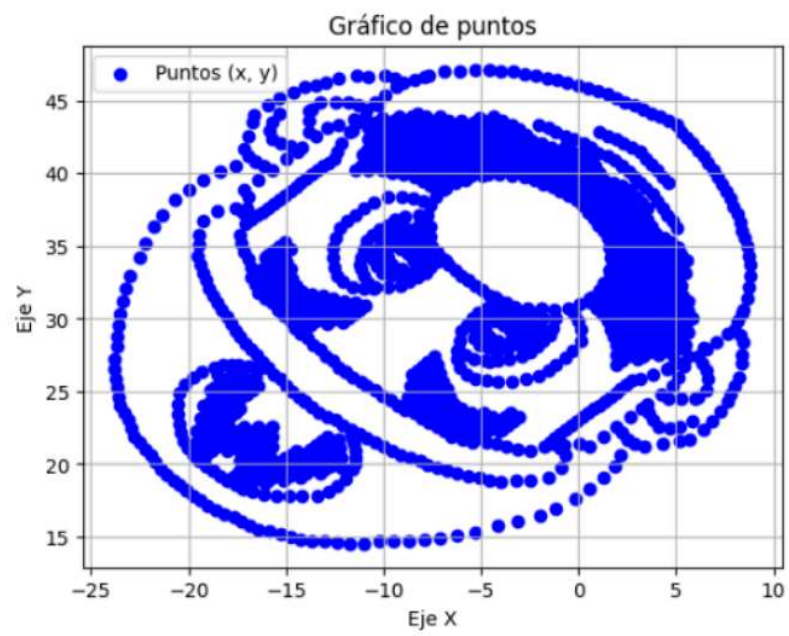


Figura 6: Gráfico de los Datos Rotados y Traslados

---

## Guardar Datos

---

Guardamos los datos ya descryptados, rotados y trasladados en nuevos archivos.

```
1 guardar_datos.save(desempaquetar_x, desempaquetar_y, "datos\valores
  descryptados.txt")
2 guardar_datos.save(valores_rotados_x, valores_rotados_y, "datos\valores
  rotados.txt")
3 guardar_datos.save(valores_trasladados_x, valores_trasladados_y, "datos\
  valores trasladados.txt")
```

Para ver los datos *descryptados* presione aquí

Para ver los datos *rotados* presione aquí

Para ver los datos *trasladados* presione aquí

---

## Conclusiones y Trabajo Futuro

El presente proyecto ha culminado con la **implementación y validación exitosa** de un sistema de descifrado y transformación de datos fundamentado en los principios del Álgebra Lineal. Mediante la correcta aplicación de la inversión de matrices y operaciones de multiplicación, se ha logrado **descifrar la totalidad de los datos** contenidos en el archivo fuente, confirmando la precisión de la clase `Cifrado()`.

Adicionalmente, se demostró la funcionalidad de las transformaciones geométricas. Las clases `Rotacion()` y `Traslacion()` ejecutaron las manipulaciones vectoriales deseadas, lo cual fue corroborado por las gráficas finales. La **integración** de todas las clases (descifrado, desempaqueado y transformaciones) permitió procesar el archivo de datos de forma **automatizada y eficiente**, cumpliendo a cabalidad con el objetivo principal del proyecto.

### Trabajo Futuro

Para expandir el alcance de este trabajo, se proponen las siguientes líneas de mejora:

- Implementar un algoritmo de cifrado más robusto que utilice métodos de Álgebra Lineal distintos al cifrado Hill básico.
  - Optimizar el manejo de la memoria con NumPy para procesar archivos de datos de mayor volumen de manera más eficiente.
- 

## Autor y Contacto

**Junior Eduardo Garniga Rojas**

*Estudiante de La Universidad Mariano Gálvez de Guatemala*

Código Fuente disponible en:

**jgarnigar**

---

## Anexo A: Estructura Completa

Proyecto-algebra-linea

- datos/
  - valores\_trasladados.txt
  - datos\_encryptados.txt
  - datos.txt
  - practica.txt
  - valores\_desencryptados.txt
  - valores\_rotados.txt

funciones/

- \_\_init\_\_.py
- app.py
- cifrado.py
- crear\_array.py
- desempaquetar.py
- ecuacion.py
- graficar.py
- guardar.py
- rotacion.py
- traslacion.py

imagenes

- datos\_desencryptados\_test.png
- datos\_desencryptados.png
- datos\_rotados\_test.png
- datos\_rotados.png
- datos\_trasladados\_test.png
- datos\_trasladados.png

.gitignore

FICHA\_TECNICA.md

INSTRUCCIONES\_PROYECTO.pdf

PROYECTO\_ALGEBRA\_LINEAL.pdf

PROYECTO\_ALGEBRA\_LINEAL.tex

README.md

clases.py

ecuacion.py

main.py

requirements.txt

---



---

## Anexo B: Herramientas y Tecnologías

Tecnología	Versión	Función en el proyecto
Python	[V 3.10]	Lenguaje de programación principal utilizado para implementar las clases de cifrado, rotación y traslación.
NumPy	[V 2.1.2]	Fundamental para la manipulación y operación eficiente de las <b>matrices</b> de $6 \times 6$ y $6 \times 1$ durante el proceso de descifrado.
Matplotlib	[V3.10.3]	Utilizada para la <b>visualización de los datos</b> , generando las gráficas de comprobación y los resultados finales (desencriptados, rotados y trasladados).

---

## Anexo C: Fundamentos Teóricos

---

### Matriz

Una matriz es un conjunto bidimensional de números o expresiones dispuesto en filas y columnas. Es la herramienta fundamental utilizada para codificar y descifrar datos en este proyecto. En nuestro caso, se emplea una matriz cuadrada de  $6 \times 6$  como clave de codificación.

### Matriz Inversa

La matriz inversa, denotada como  $A^{-1}$  para una matriz  $A$ , es crucial para el descifrado. Si  $A \cdot X = B$  (donde  $X$  es la matriz de datos encriptados y  $B$  es la matriz de datos originales), entonces la matriz original se recupera multiplicando la matriz inversa por la matriz encriptada:  $A^{-1} \cdot A \cdot X = A^{-1} \cdot B$ . Para que una matriz tenga inversa, su determinante debe ser distinto de cero.

### Cifrado Hill (Conceptual Simple)

El Cifrado Hill es un esquema de cifrado por sustitución polialfabética basado en el Álgebra Lineal. Consiste en dividir el mensaje de texto en bloques (vectores) y multiplicarlos por una **\*\*matriz clave\*\*** (nuestra matriz  $6 \times 6$ ) para obtener el texto cifrado. Para descifrar el mensaje, el texto cifrado se multiplica por la **\*\*matriz inversa\*\*** de la clave, que es el método que replicamos en este proyecto.