Matemáticas Aplicadas III

Depto. Matemáticas & Estadística / Facultad de Ingeniería

Segunda Jornada de Resolucion de Problemas (Subespacios fundamentales - Conjuntos ortonormales).

Descripción de la jornada: En esta jornada usted va a encontrar 3 problemas. Debe trabajar e intentar completar todos los enunciados. Cada problema puede ser abordado en forma grupal utilizando todos los recursos que usted disponga. La idea central es que elaboren sus respuestas o construcciones siguiendo los lineamientos dados a continuación:

1. Escriba con claridad sus propias ideas.
2. Argumente usando lenguaje propio de las matemáticas: geométrico o algebraico, según sea el caso, cada una de sus respuestas.
3. Respalde donde considere conveniente sus argumentos geométricos y algebraicos con imágenes o applets diseñadas en alguna de las herramientas que hacen parte de la mochila digital.
4. Resuelva los problemas de manera autónoma; describiendo estrategias para “atacar” el problema, poniendo a prueba sus soluciones y en lo posible planteando nuevos e “interesantes” interrogantes asociados al problema formulado. Si utiliza un resultado o teorema, este debe ser debidamente citado y si es un resultado ajeno a los estudiados en el curso o del texto guía debe proveer la respectiva demostración.
5. Respuestas sospechosas desde el punto de vista argumentativo serán evaluadas en forma oral posteriormente a la finalización de la jornada.

Su archivo *.ipynb* debe estar en la carpeta Entregas JSP 2 antes de la finalización de la sesión de clase (7:00pm). El nombre de este archivo debe tener el formato **Grupo_#_JRP2.ipynb** ```

GRUPO 2:

Isabella Huila Cerón - *A00394751*

Leidy Daniela Londoño Candelo - *A00392917*

Pablo Fernando Pineda Patiño - *A00395831*

Subespacios fundamentales de una matriz

1. (16 puntos) Considere el sistema de ecuaciones

$$A = \begin{bmatrix} 3 & 2 & -1 & 4 \\ 1 & 0 & 3 & 2 \\ -6 & -4 & 2 & -8 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 4 \\ 0 \\ 2 \end{bmatrix}.$$

   a. (4 puntos) ¿Es posible ver que tanto el conjunto de columnas de A como el de filas forman conjuntos linealmente dependiente sin realizar ninguna operación? Explique.

   Determine si el vector $\mathbf{b}$ pertenece a cada uno de los siguientes espacios: espacio columna $C(A)$, espacio fila $C(A^T)$ y espacio nulo $N(A)$. Argumente las tres respuestas.

   b. (4 puntos) Encuentre una base para los siguientes espacios fundamentales de la matriz A: $C(A)$, $N(A^T)$ y $C(A^T)$.

   c. (4 puntos) Hay dos de estos espacios que se pueden representar geométricamente en $\mathbb{R}^3$. Construya dichos subespacios en un applet de GeoGebra que incluya los vectores que los generan explícitamente. Identifique dichos subespacios con una ecuación que determine a cada uno.

   d. (4 puntos) Muestre si existe o no alguna relación geométrica entre los espacios $C(A)$ y $N(A^T)$. Teniendo en cuenta que $C(A)$ y $N(A^T)$ estan contenidos en $\mathbb{R}^k$ y que la suma $dim(C(A)) + dim(N(A^T)) = k$, ¿es posible encontrar un vector en $\mathbb{R}^k$ que no pertenezca a $C(A)$ ni a $N(A^T)$? Explique.

1a.

Podemos identificar que tanto las filas y columnas de A son linealmente dependientes sin realizar ninguna operación utilizando la idea de multiplo escalar para obtener a la otra fila del conjunto, teniendo en cuenta esto, analizamos las filas de A, observamos que al mutiplicar la primera fila por -2, obtenemos la tercera fila :

$$\begin{bmatrix} -6 & -4 & 2 & -8 \end{bmatrix} = -2 * \begin{bmatrix} 3 & 2 & -1 & 4 \end{bmatrix}$$

En el caso del conjunto de columnas, no se puede ver facilmente, ya que no se tiene un escalar que genere a otra columna

Para determinar si el vector b pertenece al espacio columna $C(A)$, debemos resolver el sistema de ecuaciones $A\mathbf{x} = \mathbf{b}$ y verificar si existe al menos una solución.

Para ellos, el sistema de ecuaciones a resolver es:

$$\begin{cases} 3x_1 + 2x_2 - x_3 + 4 = 4 \\ 1x_1 + 0 + 3x_3 + 2 = 0 \\ -6x_1 + -4x_2 + 2x_3 - 8 = 2 \end{cases}$$

Aplicaremos eliminación gaussiana para resolverlo:

```
from numpy import *
from sympy import *
from matplotlib import pyplot as plt
from numpy.linalg import norm
from scipy.linalg import lu_factor, lu_solve, qr, solve_triangular, inv, solve
```

```
A_b = Matrix([[3, 2, -1, 4, 4],
              [1, 0, 3, 2, 0],
              [-6, -4, 2, -8, 2]])

# Use sympy.rref() method
A_brref = A_b.rref()
print(f"La forma escalonada reducida de la matriz A aumentada con el vector b y el vector
```

```
La forma escalonada reducida de la matriz A aumentada con el vector b y el vector de
[1, 0,  3,  2, 0],
[0, 1, -5, -1, 0],
[0, 0,  0,  0, 1]]), (0, 1, 4))
```

Como podemos observar, al realizar la reducción gaussiana en la ultima fila nos da una inconsistencia, ya que tenemos 0 igualado a un valor, por lo tanto el vector b no pertenece la espacio columna

Para el caso del espacio fila $C(A^T)$, no es necesario verificar si el vector $\mathbf{b}$ pertenece a este espacio de forma separada, ya que esta comprobación es equivalente a verificar si $\mathbf{b}$ pertenece al espacio columna $C(A)$.

Esto se debe a que el espacio fila de una matriz A es igual al espacio columna de su matriz traspuesta A^T, y viceversa. Es decir:

$C(A^T) = C(A)^T$

Donde $C(A)^T$ representa el espacio formado por los vectores traspuestos del espacio columna $C(A)$.

Por lo tanto, si ya hemos comprobado que $\mathbf{b}$ pertenece al espacio columna $C(A)$, esto implica que $\mathbf{b}^T$ pertenece a $C(A)^T$, el cual es igual a $C(A^T)$.

Entonces, como en el espacio columna de A no tenia solucion, entonces en este caso tampoco b pertenece al espacio fila

Respuesta corregida del cuadro de arriba:

...

Para determinar si el vector $\mathbf{b}$ pertenece al espacio nulo $N(A)$ de la matriz A, debemos verificar si $A\mathbf{b} = \mathbf{0}$, donde $\mathbf{0}$ es el vector nulo.

Calculemos $A\mathbf{b}$:

$$A\mathbf{b} = \begin{bmatrix} 3 & 2 & -1 & 4 \\ 1 & 0 & 3 & 2 \\ -6 & -4 & 2 & -8 \end{bmatrix} \begin{bmatrix} 4 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 12 + 0 - 2 + 0 \\ 4 + 0 + 6 + 0 \\ -24 - 0 + 4 - 16 \end{bmatrix} = \begin{bmatrix} 10 \\ 10 \\ -36 \end{bmatrix}$$

Como $A\mathbf{b} \neq \mathbf{0}$, podemos concluir que $\mathbf{b}$ no pertenece al espacio nulo $N(A)$.

Para que un vector $\mathbf{x}$ pertenezca al espacio nulo $N(A)$, debe cumplir la ecuación $A\mathbf{x} = \mathbf{0}$. En este caso, al multiplicar A por $\mathbf{b}$, no obtenemos el vector nulo $\mathbf{0}$, sino un vector distinto de $\mathbf{0}$.

Por lo tanto, el vector $\mathbf{b}$ no satisface la condición necesaria para pertenecer al espacio nulo $N(A)$.

1b.

Para encontrar una base de $C(A)$ debemos realizar la reducción gaussiana:

```
A_b = Matrix([[3, 2, -1, 4],
              [1, 0, 3, 2],
              [-6, -4, 2, -8]])

# Use sympy.rref() method
A_brref = A_b.rref()
print(f"La forma escalonada reducida de la matriz A aumentada con el vector b y el vector
```

```
La forma escalonada reducida de la matriz A aumentada con el vector b y el vector de
[1, 0,  3,  2],
[0, 1, -5, -1],
[0, 0,  0,  0]]), (0, 1))
```

observamos que nos da un pivote en la columna 1 y 2, por lo tanto, la base de a es:

$$\text{Base } (C(A)) = \left\{ \begin{bmatrix} 3 \\ 1 \\ -6 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \\ -4 \end{bmatrix} \right\}$$

Para el caso de la base de $C(A^T)$ realizamos la reducción gaussiana:

```
A_b = Matrix([[3, 1, -6],
              [2, 0, -4],
              [-1, 3, 2],
              [4, 2, -8]])

# Use sympy.rref() method
A_brref = A_b.rref()
print(f"La forma escalonada reducida de la matriz A aumentada con el vector b y el vector
```

```
La forma escalonada reducida de la matriz A aumentada con el vector b y el vector de
[1, 0, -2],
[0, 1,  0],
[0, 0,  0],
[0, 0,  0]]), (0, 1))
```

observamos que nos da un pivote en la columna 1 y 2, por lo tanto, la base de a es:

$$\text{Base } (C(A^T)) = \left\{ \begin{bmatrix} 3 \\ 2 \\ -1 \\ 4 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 3 \\ 2 \end{bmatrix} \right\}$$

Para el caso de la base $N(A^T)$, debemos encontrar el espacio nulo, sabemos que satisface la ecuación $A\mathbf{x} = \mathbf{0}$. Por lo tanto, debemos resolver el sistema homogéneo

$$\begin{bmatrix} 3 & 1 & -6 \\ 2 & 0 & -4 \\ -1 & 3 & 2 \\ 4 & 2 & -8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Usando la forma escalonada reducida de A^T, se tiene:

$$\begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

si tomamos $x3 = t$ entonces $x1 = 2t, x2 = 0$

por lo tanto,

$$\mathbf{x} = \begin{bmatrix} 2t \\ 0 \\ t \end{bmatrix} = t \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}$$

En consecuencia, una base para el espacio nulo de A es:

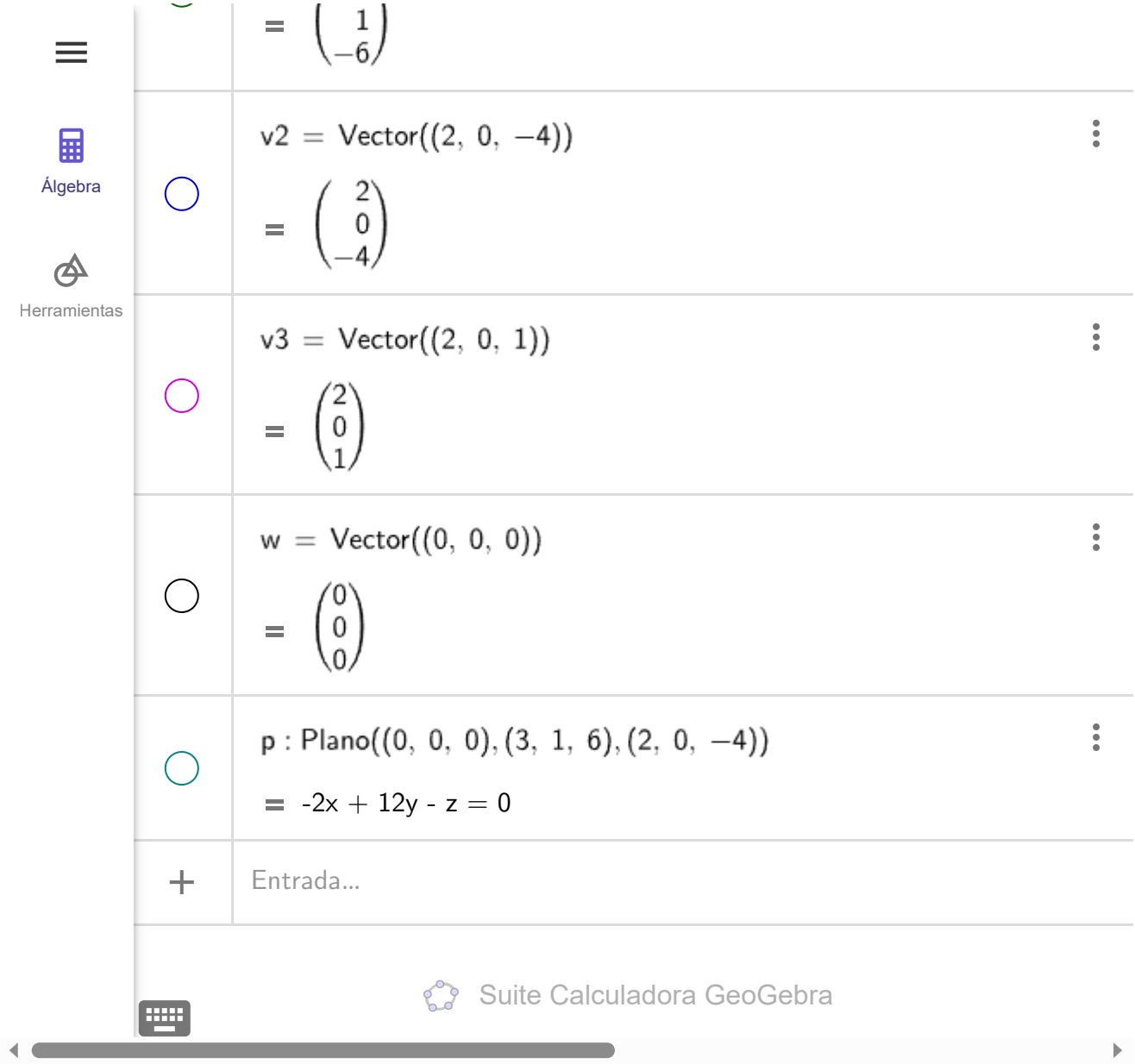
$$\text{Base } (N(A)) = \left\{ \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix} \right\}$$

1c.

```
import IPython.display as IP

iframe = IP.IFrame(
  src="https://www.geogebra.org/calculator/y3v4zu2q",
  width="1200px",
  height="600px"
  )

IP.display_html(iframe)
```



1d.

Para determinar si existe o no alguna relación geométrica entre los espacios $C(A)$ y $N(A^T)$, debemos analizar sus dimensiones y su posición relativa en el espacio $\mathbb{R}^k$.

Primero, recordemos que la dimensión de $C(A)$ es el número de columnas linealmente independientes de A, que en este caso es 2. Por otro lado, la dimensión de $N(A^T)$ es el número de vectores linealmente independientes que satisfacen la ecuación $A\mathbf{x} = \mathbf{0}$, que en este caso es 1.

Como la dimensión de $C(A)$ es mayor que la dimensión de $N(A^T)$, sabemos que $C(A)$ no puede estar contenido en $N(A^T)$. Esto se debe a que, si $C(A)$ estuviera contenido en $N(A^T)$, entonces la dimensión de $C(A)$ sería menor o igual que la dimensión de $N(A^T)$, lo cual no es el caso.

Además, como $C(A)$ y $N(A^T)$ son subespacios de $\mathbb{R}^k$, sabemos que deben estar en posiciones relativas. Por ejemplo, si $C(A)$ está contenido en $\mathbb{R}^3$, entonces $N(A^T)$ debe estar

en $\mathbb{R}^k \setminus \mathbb{R}^3$.

Por lo tanto, podemos concluir que no existe ninguna relación geométrica entre los espacios $C(A)$ y $N(A^T)$.

En cuanto a la posibilidad de encontrar un vector en $\mathbb{R}^k$ que no pertenezca a $C(A)$ ni a $N(A^T)$, debemos tener en cuenta que la suma de las dimensiones de estos dos espacios es igual a la dimensión de $\mathbb{R}^k$. Esto significa que no hay espacio para ningún otro vector que no pertenezca a alguno de estos dos espacios.

Por lo tanto, podemos concluir que no es posible encontrar un vector en $\mathbb{R}^k$ que no pertenezca a $C(A)$ ni a $N(A^T)$.

Bases y complementos ortogonales

2. (17 puntos)

   a. (3 puntos) Considere la matriz A del punto anterior. Explique por qué el algoritmo de Gram-Schmidt fallaría en encontrar una base ortogonal del subespacio generado por los cuatro vectores columna de A.

   b. (7 puntos) El conjunto de vectores S formado por los vectores de las bases de los espacios $C(A)$ y $N(A^T)$ ¿Es una base una base ortonormal para $\mathbf{R}^3$? Explique. De no ser una base ortonormal, encuentre una base ortonormal para $\mathbf{R}^3$ que incluya vectores del espacio $C(A)$ y de su complemento ortogonal, $N(A^T)$. Justifique su respuesta.

   c. (7 puntos) ¿El sistema lineal del punto 1, $A\mathbf{x} = \mathbf{b}$, tiene solución? Explique. Halle la solución por mínimos cuadrados del sistema $A\mathbf{x} = \mathbf{b}$ resolviendo el sistema de ecuaciones normales $A^T A\hat{\mathbf{x}} = A^T\mathbf{b}$ utilizando o bien la factorización LU o bien la factorización QR. Determine que tan cercana es su solución con respecto a la obtenida utilizando el comando linalg.lstsq.

2a

Tenemos en cuenta la matriz A:

$$A = \begin{bmatrix} 3 & 2 & -1 & 4 \\ 1 & 0 & 3 & 2 \\ -6 & -4 & 2 & -8 \end{bmatrix}$$

El algoritmo de Gram-Schmidt es un método para construir una base ortogonal a partir de un conjunto de vectores linealmente independientes.

El problema con la matriz A es que los vectores columna no son linealmente indenpendientes. De manera más especifica, el tercer vector columna es una combinación lineal de los otros 3, por jemplo si le mandamos los siguientes vectores:

$$\mathbf{a_1} = \begin{bmatrix} 3 \\ 1 \\ -6 \end{bmatrix}, \quad \mathbf{a_2} = \begin{bmatrix} 2 \\ 0 \\ -4 \end{bmatrix}, \quad \mathbf{a_3} = \begin{bmatrix} -1 \\ 3 \\ 2 \end{bmatrix} \quad \mathbf{a_4} = \begin{bmatrix} 4 \\ 2 \\ -8 \end{bmatrix}$$

El algoritmo va a parar justo cuando inicie la operación para a_3 y lo podemos observar en el siguente código:

```
import numpy as np



def gram_schmidt(vectors):
    # Lista para almacenar los vectores ortonormales
    q_vectors = []

    vectors = [v.astype(float) for v in vectors]
    n = len(vectors)

    for i in range(n):
        # Inicialmente se establece q como el vector a actual
        q = vectors[i]

        # Restamos la proyección de q sobre los vectores anteriores
        for j in range(i):
            q -= np.dot(vectors[i], q_vectors[j]) * q_vectors[j]

        # se comprueba si q es el vector cero
        if np.allclose(q, 0):
            print(f"El vector a_{i+1} es una combinación lineal de los vectores anteriore
            return q_vectors

        # Normalizamos q
        q /= np.linalg.norm(q)

        # Añadimos q a la lista de vectores ortonormales
        q_vectors.append(q)

    print("El conjunto de vectores es linealmente independiente.")
    return q_vectors

# iii.
# se Define el conjunto de vectores
a_1 = np.array([3, 1, -6])
a_2 = np.array([2, 0, -4])
a_3 = np.array([-1,3,2])
a_4 = np.array([4,2,-8])


vectors = [a_1, a_2, a_3, a_4]

# se aplica el algoritmo de Gram-Schmidt
q_vectors = gram_schmidt(vectors)

# Se imprimen los vectores ortonormales
for i, q in enumerate(q_vectors, 1):
    print(f"q_{i} = {q}")
```

```
 El vector a_3 es una combinación lineal de los vectores anteriores.
 q_1 = [ 0.44232587  0.14744196 -0.88465174]
 q_2 = [ 0.06593805 -0.98907071 -0.13187609]
```

2b

Como se Hallaron en el primer punto,

$$\text{Base } (C(A)) = \left\{ \begin{bmatrix} 3 \\ 1 \\ -6 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \\ -4 \end{bmatrix} \right\}$$

y tambien tenemos:

$$\text{Base } (N(A)) = \left\{ \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix} \right\}$$

Podemos comprobar con el algoritmo de Gram-Schmidt que si el conjunto de vectores es una base ortonormal. teniendo S como:

$$\text{s} = \left\{ \begin{bmatrix} 3 \\ 1 \\ -6 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \\ -4 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix} \right\}$$

```
import numpy as np

C_A = np.array([[3.0, 1.0, -6.0],
                [2.0, 0.0, -4.0]], dtype=np.float64)
N_A_T = np.array([[2.0, 0.0, 1.0]], dtype=np.float64)

# Construir el conjunto S
S = np.concatenate((C_A, N_A_T), axis=0)

print("Conjunto S:")
print(S)

# Aplicar el algoritmo de Gram-Schmidt
def gram_schmidt(vectors):
    q_vectors = []
    for v in vectors:
        q = v.copy()
        for u in q_vectors:
            q -= np.dot(v, u) * u
        if np.allclose(q, 0):
            return q_vectors
        q /= np.linalg.norm(q)
        q_vectors.append(q)
    return q_vectors

# Encuentra la base ortonormal
ortho_basis = gram_schmidt(S)

print("\nBase ortonormal para R^3:")
for i, q in enumerate(ortho_basis, 1):
    print(f"u{i} = {q}")
```

```
Conjunto S:
[[ 3.  1. -6.]
 [ 2.  0. -4.]
 [ 2.  0.  1.]]

Base ortonormal para R^3:
u1 = [ 0.44232587  0.14744196 -0.88465174]
u2 = [ 0.06593805 -0.98907071 -0.13187609]
u3 = [0.89442719 0.         0.4472136 ]
```

2c

Como se menciono anteriormente en el punto 1, el sistema lineal $A\mathbf{x} = \mathbf{b}$ no tiene solución, debido a que al final la matriz queda de la siguiente manera, por el metodo de Gauss jordan.

$$A = \begin{bmatrix} 1 & \frac{2}{3} & \frac{-1}{3} & \frac{4}{3} \\ 0 & \frac{-2}{3} & \frac{10}{3} & \frac{2}{3} \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \frac{4}{3} \\ \frac{-4}{3} \\ 10 \end{bmatrix}.$$

podemos observar que se obtiene una fila de ceros, pero que su solución es 10 lo que significa que el sistema de ecuaciones es inconsistente, lo que implica que Ax=b no tiene solución.

Vamos a solucionarlo por minimos cuadrados con factorización QR, con el siguiente código, luego lo vamos a comparar con el comando linalg.lstsq.

```
import numpy as np
from scipy import linalg
from sympy import *


A = np.array([[3, 2, -1, 4], [1, 0, 3, 2], [-6, -4, 2, -8]])
b = np.array([4, 0, 2])

#Back substitution algorithm
def backsubs(R, b):
    m, n = R.shape
    x = np.zeros(n)

    for i in range(m-1, -1, -1):
        tmp = b[i]
        for j in range(i+1, n):
            tmp -= x[j]*R[i, j]

        x[i] = tmp / R[i, i]

    return x

# Solución QR de mínimos cuadrados de sistemas lineales m x n
Q, R = np.linalg.qr(A)

Qt = Q.transpose()

y = Qt.dot(b)

z = backsubs(R,y)

print("x =",z,"\n")

print("Ax =",A.dot(z),"\n")


import numpy as np
from scipy import linalg
from sympy import *

A = np.array([[3,2,-1,4], [1,0,3,2], [-6,-4,2,-8]])

b = np.array([4, 0, 2])
print("A =",A,"\n")
print("b =",b,"\n")

# Solución por mínimos cuadrados de sistemas lineales m x n
x = np.linalg.lstsq(A, b, rcond=None)[0]

print("x =",x,"\n")

print("Ax =",A.dot(x),"\n")
```

Podemos ver que con el código del comando linalg.lstsq. y el código con la factorización QR son diferentes por lo tanto, la solución no es cercana.

Sistemas de ecuaciones lineales mal condicionados

3. (17 puntos) Para responder las siguientes preguntas usted debe construir un código que genere la matriz $n \times n$, $H = I_n - 2ww^T$, donde el n-vector w es unitario, $w^T w = 1$ (matriz de Householder). Además, debe calcular un $n-$vector $\mathbf{b}$ multiplicando la matriz H por el $n-$vector de unos (np.ones(n)), para valores de n de 5 en 5 empezando en $n = 5$ hasta $n = 50$. Es decir, usted debe generar al menos 10 sistemas de ecuaciones lineales cuadrados de distinto tamaño cuya solución es el vector $\mathbf{x} = ones(n)$.

   a. (8 puntos) *Solución de sistemas de ecuaciones lineales.* Su código debe resolver cada sistema de tres formas distintas: una con la matriz inversa, $\mathbf{x} = A^{-1}\mathbf{b}$, otra con la factorización QR y una última con la función linalg.solve. La función linalg.solve no debe usarse en los otros dos métodos de solución. No necesita mostrar (imprimir en pantalla) ni la matriz H, ni el vector solución.

   b. (5 puntos) *Sistemas mal condicionados.* ¿Teóricamente se podría afirmar que para cada tamaño de problema n el sistema $A\mathbf{x} = \mathbf{b}$ tiene solución? Su implementación debe imprimir en pantalla un mensaje indicando para qué tamaño n el sistema tiene una solución numérica incorrecta. Usted debe sustentar por qué en algunos casos a pesar que numéricamente se llega a una solución, esta no es correcta.

   c. (4 puntos) *Grafico de precisión.* Investigue como calcular el error relativo entre vectores y úselo como una medida del error entre su solución $\tilde{\mathbf{x}}$ y la solución real $\mathbf{x} = \mathbf{ones(n)}$ del problema para cada tamaño n. Los resultados debe utilizarlos para hacer dos gráficas con matplotlib. Una que le permita comparar la precisión de los tres métodos y otra solo con los resultados de la factorización QR y linalg.solve. El eje de abcisas debe corresponder al tamaño del problema n.

```
import numpy as np
from numpy.linalg import inv, qr
from scipy.linalg import solve
import matplotlib.pyplot as plt


'''
np.eye(n) crea matriz de identidad n x n
por ejemplo si n=3 entonces genera esta matriz
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]

'''

#2
def generarMatriz(n):
    w = np.random.randn(n)
    w = w / np.linalg.norm(w)
    H = np.eye(n) - 2 * np.dot(w, w.T)
    b = H @ np.ones(n)
    return H, b, w


#a
def resolvSis(H, b):
    x_inv = inv(H) @ b

    Q, R = np.linalg.qr(H)
    R_inv = np.linalg.inv(R)
    x_qr = R_inv @ (Q.T @ b)

    x_solve = solve(H, b)[0]
    return x_inv, x_qr, x_solve


#b
```