

ÁLGEBRA LINEAL COMPUTACIONAL

2do Cuatrimestre 2025

Laboratorio N° 03: Normas y condición de matrices.

1. Normas

Sea $x \in \mathbb{R}^n$. Se define la norma $p \in \mathbb{N}$ de x como:

$$\|x\|_p = \left(\sum_{i=1}^n |x_i|^p \right)^{1/p}$$

Ejercicio 1 (Puntos de norma 1). Desarrolle:

- Una función `norma(x,p)` que reciba un vector x (un objeto iterable) y una norma p y retorne su norma.
- Una función `normaliza(X,p)` que reciba una lista de vectores X y una norma p y retorne una lista Y donde cada elemento corresponde a normalizar los elementos de X con la norma p .

Emplee la función construida para graficar, en un mismo plot, los vectores con norma 1 de \mathbb{R}^2 según las normas $p = 1, 2, 5, 10, 100, 200$. Describa las figuras que observa. Identifique a qué forma funcional converge la norma para valores de $p \rightarrow +\infty$. Muestre gráficamente que ese límite puede expresarse con la norma infinito y

- Modifique la función `norma(x,p)` para que pueda recibir el argumento `p='inf'` con

$$\|x\|_\infty = \max_{i=1,\dots,n} |x_i|$$

- Incorpore la norma infinito al gráfico anterior.
- ¿En qué valores coinciden todas las normas graficadas?

2. Normas matriciales inducidas

La norma q, p inducida de una matriz $A \in \mathbb{R}^{m \times n}$ se define como

$$\|A\|_{q,p} = \max_{x/\|x\|_p=1} \|Ax\|_q$$

para dos normas $\|\cdot\|_p : \mathbb{R}^n \rightarrow \mathbb{R}^+$ y $\|\cdot\|_q : \mathbb{R}^m \rightarrow \mathbb{R}^+$. Noten que las normas no tienen por qué ser idénticas, ni la matriz A necesita ser cuadrada.

Ejercicio 2 (Normas q, p). En este ejercicio desarrollaremos funciones aleatorias y exactas para calcular las normas inducidas de la matriz.

- a) Aplicamos el método de *Monte Carlo* para estimar la norma inducida de una matriz. Para ello, implementar una función `normaMatMC(A,q,p,Np)` que estime $\|A\|_{q,p}$ numéricamente, usando Np vectores x de \mathbb{R}^n generados al azar. La función debe retornar la norma $\|A\|_{q,p}$ y el vector x en el cual se alcanza el máximo.
- b) Use la función construida para estimar las siguientes normas, junto a las posiciones en las que se alcanzan:

- | | |
|---|--|
| i. $\ I\ _{2,1}, I \in \mathbb{R}^{2,2}$ | v. $\ A\ _{2,2}, A = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$ |
| ii. $\ I\ _{1,2}, I \in \mathbb{R}^{2,2}$ | vi. $\ A\ _{2,2}, A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ |
| iii. $\ I\ _{2,\infty}, I \in \mathbb{R}^{2,2}$ | vii. $\ A\ _{\infty,\infty}, A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ |
| iv. $\ I\ _{\infty,2}, I \in \mathbb{R}^{2,2}$ | viii. $\ A\ _{2,\infty}, A = \begin{pmatrix} 10 & 10 \\ 0 & 0 \end{pmatrix}$ |

donde I es la matriz identidad.

En todos los casos, calcule 100 veces la norma, usando $Np=1000$, de forma de generar una muestra de la misma. ¿Siempre ese estable el valor que obtiene? ¿Por qué? Interprete geoméricamente los resultados.

- c) Construya una función `normaExacta(A,p=[1,'inf'])` que calcule las normas 1 e ∞ de una matriz A usando las expresiones:

$$\|A\|_{\infty} = \max_{1 \leq i \leq n} \sum_{j=1}^n |A_{ij}| \quad \|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^n |A_{ij}|$$

y compárelas con lo obtenido en el punto anterior. *Nota: cuando las normas de partida y llegada son las mismas ($p = q$) la norma $\|A\|_{p,p}$ de nota simplemente $\|A\|_p$*

3. Condicionamiento de matrices

Una matriz $A \in \mathbb{R}^{n \times n}$ se dice bien condicionada si al resolver el problema

$$Ax = b$$

pequeñas variaciones en el valor de b no producen grandes variaciones en la solución x que encontramos. En este contexto, resulta útil el número de condición

$$\kappa_p(A) = \|A\|_p \|A^{-1}\|_p \quad (1)$$

que permite dar una cota a cómo se propaga el error cometido debido a una pequeña variación en el vector de soluciones:

$$\frac{\|x - \tilde{x}\|_p}{\|x\|_p} \leq \kappa_p(A) \frac{\|b - \tilde{b}\|_p}{\|b\|_p}$$

donde $Ax = b$ y $A\tilde{x} = \tilde{b}$.

Ejercicio 3 (Número de condición). Desarrolle las funciones:

- a) `condMC(A,p)` que estime el número de condición de A usando la norma inducida p . La función debe emplear la función `normaMatMC` desarrollada anteriormente¹.
- b) Desarrolle una rutina para poner a prueba la cota dada por el número de condición. Para esto:
- Desarrolle una función `variaPerc(b,perc)` que reciba un vector $b \in \mathbb{R}^n$ y varíe sus elementos en un porcentaje aleatorio. Identifique qué norma es más sensible a esta variación. *Tip:* Una forma de realizar esto es multiplicar cada elemento de b por un número aleatorio entre $1 - \text{perc}/100$ y $1 + \text{perc}/100$. De esta forma, tendremos que $\|b - \tilde{b}\|/\|b\| \leq \text{perc}/100$.
 - Construya una función que reciba a una matriz A , una norma p , un porcentaje `perc` y una cantidad de realizaciones `Np` y siga el siguiente esquema para encontrar la peor combinación de b y \tilde{b} para la matriz A :

- genera `Np` instancias de b y `Np` instancias de \tilde{b} , donde la variación porcentual es a lo sumo `perc`.
- resuelve $Ax = b$ y $A\tilde{x} = \tilde{b}$.
- Calcula $e_x = \|x - \tilde{x}\|/\|x\|$ y $e_b = \|b - \tilde{b}\|/\|b\|$
- Calcula el número de condición de A .
- Retorna en una lista el número de condición, los $2Np$ errores de e_x y e_b , y los valores de b y \tilde{b} en los que se alcanza el máximo de e_x

Empleando esta función, grafiquen un histograma de $\frac{\|x - \tilde{x}\| \cdot \|b\|}{\|x\| \cdot \|b - \tilde{b}\|}$ marcando con una línea vertical el número de condición de cada una de las matrices A presentadas a continuación, para las normas 1, 2 e infinito, y porcentajes `perc=1, 5, 10`. ¿Cuán cercanos son estos valores al número de condición?

$$\begin{array}{ll} \blacksquare A = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} & \blacksquare A = \begin{pmatrix} 1000 & 1/1000 \\ 0 & 1000 \end{pmatrix} \\ \blacksquare A = \begin{pmatrix} 501 & 499 \\ 500 & 500 \end{pmatrix} & \blacksquare A = \begin{pmatrix} 1/1000 & 1000 \\ 0 & 1000 \end{pmatrix} \end{array}$$

- Usando la función del punto anterior, realice 10 búsquedas de vectores b y \tilde{b} en los cuales el error $\|x - \tilde{x}\|/\|x\|$ es máximo para cada una de las normas y porcentajes de error considerados. Use `Np=100000` muestras para cada una de las matrices listadas en el punto anterior. Compute $\Delta b = \tilde{b} - b$ y grafique b y Δb usando la función `arrow` de `matplotlib`, empleando una flecha para mostrar b y otra que indique Δb partiendo desde b . ¿Cuán estables son los vectores obtenidos para cada matriz y porcentaje? Compare como varía el resultado obtenido al aumentar `perc` y la definición de norma. ¿Varía la orientación de Δb con `perc`?

¹Nota: en el contexto de esta práctica, aún no disponemos de una función `inversa(A)` propia. Por esto, pueden emplear `numpy.linalg.inv` momentáneamente para el cálculo de la inversa. Al entregar el módulo ALC deben reemplazar estas instancias por las funciones desarrolladas en el curso.

Ejercicios extra

Ejercicio 4 (Optimización por Luus-Jaakola). A lo largo de este labo, deberían haberse preguntado sobre los tiempos de corrida para los análisis realizado. En particular, probar en N_p valores al azar es muy ineficiente: si queremos encontrar un mínimo o máximo dado que las funciones son continuas debería haber una estrategia mejor. Si bien más adelante vamos a programar metodologías más directas, un pequeño paso a esta altura nos puede ahorrar muchos cálculos.

Programen un buscador de Luus-Jaakola para mejorar la función del punto 2.a `normaMatLJ(A,q,p,Np,maxiter=1000,variacion=2,rate=0.95)`. Se agregan los parámetros `maxiter`, que limita el número máximo de iteraciones del algoritmo y `variacion` que indica la variabilidad de los puntos generados. El algoritmo se inicia generando N_p vectores de norma unitaria de acuerdo a la norma p y un contador `iter`. Tomamos como x_0 al elemento de esta primer muestra tal que $\|Ax_0\|_q$ es máximo. Luego repetimos hasta llegar al máximo de iteraciones:

- Generamos N_p muestras centradas en x_0 con valores uniformemente distribuidos entre $x_0 + \text{variacion}$ y $x_0 - \text{variacion}$. (`np.random.uniform`).
- Se normalizan los nuevos vectores generados usando la norma p .
- Basandonos en la muestra de vectores normalizados:
 - Si existe un x_1 en la nueva muestra tal que $\|Ax_1\|_q > \|Ax_0\|_q$, entonces reemplazamos $x_1 = x_0$
 - Si se mantiene que $\|Ax_0\|_q$ es el máximo encontrado, hacemos `variacion = variacion*rate`
- Avanzamos un paso `iter += 1`.

Módulo ALC

Para el módulo ALC, deben programar:

```
1 def norma(x, p):
2     """
3     Devuelve la norma p del vector x.
4     """
5
6 def normaliza(X, p):
7     """
8     Recibe X, una lista de vectores no vacíos, y un escalar p. Devuelve
9     una lista donde cada elemento corresponde a normalizar los
10    elementos de X con la norma p.
11    """
12
13 def normaMatMC(A, q, p, Np):
14     """
15     Devuelve la norma ||A||_{q,p} y el vector x en el cual se alcanza
16     el máximo.
17     """
18
19 def normaExacta(A, p=[1, 'inf']):
20     """
21     Devuelve una lista con las normas 1 e infinito de una matriz A
22     usando las expresiones del enunciado 2.(c).
23     """
24
25 def condMC(A, p):
26     """
27     Devuelve el número de condición de A usando la norma inducida p.
28     """
29
30 def condExacto(A, p):
31     """
32     Que devuelve el número de condición de A a partir de la fórmula de
33     la ecuación (1) usando la norma p.
34     """
```

Tests utilizando la función `assert`:

```
1 # Tests norma
2 assert(np.allclose(norma(np.array([1,1]),2),np.sqrt(2)))
3 assert(np.allclose(norma(np.array([1]*10),2),np.sqrt(10)))
4 assert(norma(np.random.rand(10),2)<=np.sqrt(10))
5 assert(norma(np.random.rand(10),2)>=0)
6
7 # Tests normaliza
8 assert([np.allclose(norma(x,2),1) for x in normaliza([np.array([1]*
9 k) for k in range(1,11)],2) ])
10 assert([not np.allclose(norma(x,2),1) for x in normaliza([np.array
11 ([1]*k) for k in range(1,11)],1) ])
12 assert([np.allclose(norma(x,'inf'),1) for x in normaliza([np.random
13 .rand(k) for k in range(1,11)], 'inf') ])
14
15 # Tests normaExacta
16
17 assert(np.allclose(normaExacta(np.array([[1,-1],[-1,-1]]),1),2))
18 assert(np.allclose(normaExacta(np.array([[1,-2],[-3,-4]]),1),7))
19 assert(np.allclose(normaExacta(np.array([[1,-2],[-3,-4]]), 'inf'),6)
20 )
21 assert(normaExacta(np.array([[1,-2],[-3,-4]]),2) is None)
22 assert(normaExacta(np.random.random((10,10)),1)<=10)
```

```

19 assert(normaExacta(np.random.random((4,4)), 'inf') <= 4)
20
21 # Test normaMC
22
23 nMC = normaMatMC(A=np.eye(2), q=2, p=1, Np=100000)
24 assert(np.allclose(nMC[0], 1, atol=1e-3))
25 assert(np.allclose(np.abs(nMC[1][0]), 1, atol=1e-3) or np.allclose(np
    .abs(nMC[1][1]), 1, atol=1e-3))
26 assert(np.allclose(np.abs(nMC[1][0]), 0, atol=1e-3) or np.allclose(np
    .abs(nMC[1][1]), 0, atol=1e-3))
27
28 nMC = normaMatMC(A=np.eye(2), q=2, p='inf', Np=100000)
29 assert(np.allclose(nMC[0], np.sqrt(2), atol=1e-3))
30 assert(np.allclose(np.abs(nMC[1][0]), 1, atol=1e-3) and np.allclose(
    np.abs(nMC[1][1]), 1, atol=1e-3))
31
32 A = np.array([[1, 2], [3, 4]])
33 nMC = normaMatMC(A=A, q='inf', p='inf', Np=1000000)
34 assert(np.allclose(nMC[0], normaExacta(A, 'inf'), rtol=2e-1))
35
36 # Test condMC
37
38 A = np.array([[1, 1], [0, 1]])
39 A_ = np.linalg.solve(A, np.eye(A.shape[0]))
40 normaA = normaMatMC(A, 2, 2, 10000)
41 normaA_ = normaMatMC(A_, 2, 2, 10000)
42 condA = condMC(A, 2, 10000)
43 assert(np.allclose(normaA[0]*normaA_[0], condA, atol=1e-3))
44
45 A = np.array([[3, 2], [4, 1]])
46 A_ = np.linalg.solve(A, np.eye(A.shape[0]))
47 normaA = normaMatMC(A, 2, 2, 10000)
48 normaA_ = normaMatMC(A_, 2, 2, 10000)
49 condA = condMC(A, 2, 10000)
50 assert(np.allclose(normaA[0]*normaA_[0], condA, atol=1e-3))
51
52 # Test condExacta
53
54 A = np.random.rand(10, 10)
55 A_ = np.linalg.solve(A, np.eye(A.shape[0]))
56 normaA = normaExacta(A, 1)
57 normaA_ = normaExacta(A_, 1)
58 condA = condExacta(A, 1)
59 assert(np.allclose(normaA*normaA_, condA))
60
61 A = np.random.rand(10, 10)
62 A_ = np.linalg.solve(A, np.eye(A.shape[0]))
63 normaA = normaExacta(A, 'inf')
64 normaA_ = normaExacta(A_, 'inf')
65 condA = condExacta(A, 'inf')
66 assert(np.allclose(normaA*normaA_, condA))

```