Curso de Métodos Numéricos (DEMAT)

Tarea 5

Descripción:	Fechas
Fecha de publicación del documento:	**Septiembre 16, 2023**
Fecha límite de entrega de la tarea:	**Septiembre 24, 2023**

Indicaciones

Puede escribir el código de los algoritmos que se piden en una celda de este notebook o si lo prefiere, escribir las funciones en un archivo `.py` independiente e importar la funciones para usarlas en este notebook. Lo importante es que en el notebook aparezcan los resultados de la pruebas realizadas y que:

- Si se requieren otros archivos para poder reproducir los resultados, para mandar la tarea cree un archivo ZIP en el que **incluya el notebook** y los archivos adicionales.
- Si todos los códigos que se requieren para reproducir los resultados están en el notebook, no hace falta comprimir el notebook y puede anexar este archivo en la tarea del Classroom.
- Exportar el notebook a un archivo PDF y anexarlo en la tarea del Classroom como un archivo independiente. **No incluya el PDF dentro del ZIP**, porque la idea que lo pueda accesar directamente para poner anotaciones y la calificación de cada ejercicio.

Ejercicio 1 (5 puntos)

En el algoritmo de eliminación Gaussiana vimos que se puede ver como ir premultiplicando a la matriz $\mathbf{A}$ del sistema de ecuaciones por matrices triangulares inferiores elementales de la forma:

$$
\mathbf{L}_k = \begin{bmatrix}
1 & 0 & \cdots & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 \\
0 & 1 & & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 \\
\vdots & & \ddots & & \vdots & \vdots & & & \vdots & \vdots \\
\vdots & & & \ddots & \vdots & \vdots & & & \vdots & \vdots \\
0 & 0 & \cdots & 0 & 1 & 0 & \cdots & \cdots & 0 & 0 \\
0 & 0 & \cdots & 0 & -l_{k+1,k} & 1 & & & 0 & 0 \\
\vdots & \vdots & & \vdots & \vdots & & \ddots & & & \vdots \\
\vdots & \vdots & & \vdots & \vdots & & & \ddots & & \vdots \\
0 & 0 & \cdots & 0 & -l_{n-1,k} & 0 & & & 1 & 0 \\
0 & 0 & \cdots & 0 & -l_{n,k} & 0 & \cdots & \cdots & 0 & 1
\end{bmatrix},
$$

$$
l_{ik} = \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}, \quad i = k+1, k+2, \ldots, n,
$$

donde $a_{ij}^{(k-1)}$ son los elementos de la matriz $\mathbf{A}^{(k-1)} = \mathbf{L}_{k-1} \cdots \mathbf{L}_1 \mathbf{A}$.

Note que $\mathbf{L}_k = \mathbf{I} - \mathbf{l}_k \mathbf{e}_k^\top$, donde $\mathbf{I}$ es la matriz identidad de tamaño n, con el vector $\mathbf{l}_k = (0, 0, \ldots, 0, l_{k+1,k}, l_{k+2,k}, \ldots, l_{n,k})^\top$ y $\mathbf{e}_k$ es el k-ésimo vector canónico con un 1 en la posición k-ésima.

1. Muestre que la inversa de $\mathbf{L}_k$ es $\mathbf{L}_k^{-1} = \mathbf{I} + \mathbf{l}_k \mathbf{e}_k^\top$. Así, la inversa de una matriz triangular inferior elemental es otra matriz triangular inferior elemental.
2. Muestre que $\mathbf{L}_{k-1}^{-1} \mathbf{L}_k^{-1} = \mathbf{I} + \mathbf{l}_{k-1} \mathbf{e}_{k-1}^\top + \mathbf{l}_k \mathbf{e}_k^\top$, y a partir de esto, muestre que

$$
\mathbf{L} = \mathbf{L}_1^{-1} \mathbf{L}_2^{-1} \cdots \mathbf{L}_{n-1}^{-1} = \mathbf{I} + \sum_{k=1}^{n-1} \mathbf{l}_k \mathbf{e}_k^\top,
$$

por lo que $\mathbf{L}$ es una matriz triangular inferior con $1's$ en la diagonal.

Solución:

Puede escribir la solución en la celda o escribir el desarrollo en papel y tomar fotos y agregarlas al notebook. Sólo asegúrese que se vea clara la respuesta en las fotos.

Double-click (or enter) to edit

```
1. Notemos que
$$\L_k(\I + \l_k\e_k^\top) = (\I - \l_k\e_k^\top)(\I +
\l_k\e_k^\top) = \I^2 + \l_k\e_k^\top - \l_k\e_k^\top
- (\l_k\e_k^\top)^2 = \I^2 - (\l_k\e_k^\top)^2,$$
por lo cual basta ver que $(\l_k\e_k^\top)^2 = 0.$
Para ello notemos que
```

1. Notemos que
$$\mathbf{L}_k(\mathbf{I} + \mathbf{l}_k\mathbf{e}_k^\top) = (\mathbf{I} - \mathbf{l}_k\mathbf{e}_k^\top)(\mathbf{I} + \mathbf{l}_k\mathbf{e}_k^\top) = \mathbf{I}^2 + \mathbf{l}_k\mathbf{e}_k^\top - \mathbf{l}_k\mathbf{e}_k^\top - (\mathbf{l}_k\mathbf{e}_k^\top)^2 = \mathbf{I}^2 - (\mathbf{l}_k\mathbf{e}_k^\top)^2,$$
por lo cual basta ver que $(\mathbf{l}_k\mathbf{e}_k^\top)^2 = 0.$ Para ello notemos que

Ejercicio 1 (5 puntos)

Programar el algoritmo de factorización LU con pivoteo parcial y probarlo resolviendo un sistema de ecuaciones lineales de acuerdo al Algoritmo 1 de la clase 11.

1. Programe la función `factorizacionLU` que calcula la factorización LU de acuerdo al Algoritmo 1:

Entrada:

- la matriz $\mathbf{A}$ de tamaño n,
- una tolerancia τ

Salida:

- El arreglo $\mathbf{p}$ de enteros de tamaño n que tiene la información de una permutación realizada a las filas de la matriz, resultado del pivoteo parcial.
- Las matrices $\mathbf{L}$ y $\mathbf{U}$ de tamaño n.
- Un booleano que es igual a `True` si el algoritmo concluye de manera exitosa, o es `False` en caso contrario.

Nota 1: Hay que tener cuidado al escribir el código porque el algoritmo está descrito de modo que los índices de las matrices y vectores empiezan en 1, mientras que en Python empiezan en 0.

Nota 2: Puede usar el producto exterior implementado en la función [numpy.outer]

`------------`

2. Escriba la función `calcularSolucionLU` que resuelve el sistema $\mathbf{LUx} = \mathbf{Pb}$, donde $\mathbf{L}$ es una matriz triangular inferior y $\mathbf{U}$ es una matriz triangular superior.

Entrada:

- Las matrices $\mathbf{L}$ y $\mathbf{U}$
- El vector $\mathbf{b}$,
- El arreglo de enteros $\mathbf{p}$,
- Una tolerancia τ.

Salida:

- El arreglo $\mathbf{x}$ o `None`

La función debe hacer lo siguiente:

- Crear un arreglo $\hat{\mathbf{b}} = (\hat{b}_1, \ldots, \hat{b}_n)^\top$ con los elementos de $\mathbf{b} = (b_1, \ldots, b_n)^\top$ reordenados de acuerdo a $\mathbf{p} = (p_1, \ldots, p_n)^\top$, esto es, $\hat{b}_i = b_{p_i}$.
- Use las funciones `backwardSubstitution` y `forwardSubstitution` de la Tarea 4 para resolver el sistema de ecuaciones. Si no hay ningún problema, la función debe devolver el arreglo de la solución $\mathbf{x}$. En caso contrario, devolver `None`.

`------------`

3. Escriba la función `resuelverSistemaLineal` que resuelve el sistema de ecuaciones $\mathbf{Ax} = \mathbf{b}$ usando la factorización LU.

Entrada:

- La matriz $\mathbf{A}$
- El vector $\mathbf{b}$
- Una tolerancia τ

Salida:

- El arreglo $\mathbf{x}$ o `None`.

La función debe hacer lo siguiente:

- Usar la función `factorizacionLU` para obtener la factorizaci'on LU de la matriz $\mathbf{A}$.
- Si no se logró factorizar la matriz, devuelva `None`.
- En caso contrario, use la función `calcularSolucionLU` del punto anterior para resolver el sistema de ecuaciones $\mathbf{LUx} = \mathbf{Pb}$, y devolver la solución $\mathbf{x}$ o `None` según el resultado de la función.

4. Pruebe la función anterior usando los datos en el archivo `datosTarea05.zip` dentro de la carpeta `datosLU`:

Matriz:	Vector:
matA005.npy	vecb005.npy
matA010.npy	vecb010.npy
matA015.npy	vecb015.npy
matA100.npy	vecb100.npy
matA500.npy	vecb500.npy

Para cada ejemplo haga lo siguiente:

- Lea los archivos para crear la matriz $\mathbf{A}$ y el vector $\mathbf{b}$.
- Imprima el tamaño de la matriz.
- Aplique la función `resuelverSistemaLineal` usando como tolerancia $\tau = \sqrt{\epsilon_m}$, donde ϵ_m es el épsilon de la máquina.
- En caso de que sí fue obtenida la solución, imprima las primeras y últimas entradas del vector solución e imprima el valor del error $\|\mathbf{Ax} - \mathbf{b}\|$.
- Si no se obtuvo la solución, imprima un mensaje que indique eso.

Solución:

```
# Código de la función
import numpy as np

def factorizacionLU(A: np.ndarray, t: float):
    d = A.shape[0]
    L = np.eye(d)
    U = A.copy()
    p = np.arange(1, d+1)
    for k in range(d-1):
        u_rk = max(enumerate(abs(A[k:,k])), key=lambda x: x[1])
        if u_rk[1] < t:
            return p, L, U, False
        if (r:= u_rk[0] + k) != k:
            U[k, r] = U[r, k]
            p[k] = p[r]
            if k>0:
                pass
        print(u_rk )
```

```
# Pruebas

A = np.load("/content/matA005.npy")
print(A, (A[1:, 1]))
factorizacionLU(A, .4)

if (r:= 1+2) != 1:
    print(r)

for a, b in enumerate(A[1:, 1]):
    print(a, b)
```

```
[[-2.17  -0.013  1.412  1.22  -2.25 ]
 [-2.604  1.001 -0.9    1.83  -2.83 ]
 [-0.434  0.572  0.776 -1.95  -2.55 ]
 [ 0.434 -2.34  -2.34   3.62   4.81 ]
```

```
 [-4.34   1.3    1.4   -1.2   -1.1  ]] [ 1.001  0.572 -2.34   1.3  ]
(4, 4.34)
(2, 2.34)
(1, 2.3400000000000003)
(0, 3.62)
3
0 1.0010000000000001
1 0.5720000000000001
2 -2.34
3 1.3
```