Curso de Métodos Numéricos (DEMAT)

Tarea 6

Descripción:	Fechas
Fecha de publicación del documento:	**Octubre 1, 2023**
Fecha límite de entrega de la tarea:	**Octubre 9, 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 (4 puntos)

Programe el método de Gauss-Seidel para resolver el sistema de ecuaciones lineales $\mathbf{A}\mathbf{x} = \mathbf{b}$ cuando la matriz $\mathbf{A}$ es tridiagonal.

1. Escribir la función para resolver el sistema de ecuaciones:

Entradas de la función:

- la matriz $\mathbf{A}$,
- el vector $\mathbf{b}$,
- un arreglo inicial $\mathbf{x}_k$,
- el máximo número de iteraciones N y
- una tolerancia $\tau > 0$.

Descripción:

- Defina una tolerancia $\delta = \epsilon_m^{2/3}$.
- Iniciar las N iteraciones del algoritmo de Gauss-Seidel.
- Dentro del ciclo, calcular el error $E = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|$.
- Si $E < \tau$, terminar las iteraciones.
- En caso contrario, actualizar las entradas del vector $\mathbf{x}_k$ de acuerdo a la fórmula de Gauss-Seidel, sabiendo que la matriz $\mathbf{A}$ es tridiagonal para sólo hacer las operaciones con las entradas no cero de la matriz.
- Terminar las iteraciones cuando al calcular un cociente num/den, se tenga que el denominador cumple con $|den| < \delta$, para evitar problemas de dividir por cantidades muy pequeñas.

Salida de la función:

- $\mathbf{x}_k$,
- el número de iteraciones realizadas k y
- el error E.

2. Use los datos del archivo `datosTarea06.zip` para probar el algoritmo.

Matriz:	Vector:
matTridiag005.npy	vecb005
matTridiag010.npy	vecb010
matTridiag500.npy	vecb500

Pruebe primero el algoritmo haciendo pocas iteraciones para ver si el algoritmo puede converger. Esto lo ve viendo como se comporta el error E.

Si ve que es posible que el algoritmo puede converger, de un valor apropiado para N para que pueda converger el algoritmo.

En cualquier caso, imprima la primera y las últimas tres entradas del vector $\mathbf{x}_k$, el número de iteraciones realizadas k y el error E.

Solución:

```
# Funcion
import numpy as np
from numpy.linalg import norm



def gauss_seidel_tri(A: np.ndarray, b: np.ndarray, x0: np.ndarray, N: int, t: float):
    """Implementacion de Gauss-Seidel para el caso de matrces tridiagonales"""
    x = x0.copy()
    for k in range(N):
        if (E:=norm(A @ x - b)) < t:
            return x, k, E

        x[0] = (b[0] - A[0, 1]*x[1])/A[0, 0]
        for i in range(1, A.shape[0] -1):
            x[i] = (b[i] - A[i, i-1]*x[i-1] - A[i, i+1]*x[i+1])/A[i,i]
        x[-1] = (b[-1] - A[-1,-2]) / A[-1, -1]
    return x, N, norm(A @ x - b)

```

```
from numpy.core.multiarray import ndarray
import numpy as np
from numpy.linalg import norm

def gauss_seidel(A: ndarray, b: ndarray, x0: ndarray, N: int, t: float):
    """implementaciones generales de Gauss Seidel"""
    x=np.zeros(b.size)
    for k in range(N):
        if (E:=norm(A @ x - b)) < t:
            return x, k, E
        for i in range(b.size):
            if abs(A[i,i]) < t:
                return x, k, E
            x[i] = (b[i]-A[i, :i]@x[:i] - A[i, i+1:]@x0[i+1:])/A[i, i]
        # print(x0)
        x0 = x.copy()
    return x, N, norm(A @ x - b)

def gauss_seidel_2(A: ndarray, b: ndarray, x0: ndarray, N: int, t: float):
    B = A.copy()
    for i in range(A.shape[0]):
        B[i ,i] = 0
    x=np.zeros(b.size)
    for k in range(N):
        if (E:=norm(A @ x - b)) < t:
            return x, k, E
        for i in range(b.size):
            if abs(A[i,i]) < t:
                return x, k, E
            x[i] = (b[i]-B[i]@x)/A[i, i]

    return x, N, norm(A @ x - b)
```

```
# Pruebas

files = ['005', '010', '500']
matrices = (f"/content/matTridiag{mat}.npy" for mat in files)
vectores = (f"/content/vecb{vec}.npy" for vec in files)

for mat, vec in zip(matrices, vectores):
    A = np.load(mat)
    b = np.load(vec)
    x0 = np.random.rand(b.size)
    x, N, E = gauss_seidel(A, b, x0, 10, np.finfo(float).eps**(2/3))
    print(f"La sol es: {x} y el error es: {E}")


```

```
 -7.83041434e-01 -1.91308263e+00 -2.60128326e-02 -6.62620482e-01
  1.04600568e+00 -8.70902093e-02  7.51499007e-02 -7.21328256e-01
  1.24698653e+00  6.52883983e-01  1.94210740e+00 -3.98958175e-02
  2.88842835e-01 -1.18521664e+00  8.51934916e-01 -6.51262454e-01
  2.25634646e-01 -3.28914867e-01  1.33294530e-01 -1.33802741e+00
  1.00771391e+00  1.72661641e-02  2.70392576e+00 -7.19598849e-01
 -3.12760015e-01 -2.54882206e+00 -8.27532673e-01  3.11797213e-01
  1.35392910e+00  3.16928836e-01  5.47662893e-01 -1.26072912e+00
  7.01786967e-02  1.48920528e-01  1.18622762e+00  6.21537848e-01
  1.14794455e+00 -8.13328457e-01 -1.57612042e-01 -2.83937817e-01
  1.70350366e+00  2.37241370e-01  4.48597507e-01 -1.57345456e+00
 -1.28403518e+00 -1.34315587e+00  1.10652393e+00 -1.22073473e-02
  7.50927424e-01 -7.06384537e-01  4.39326671e-02  5.12078524e-01
  8.01219165e-01 -4.71651691e-02  6.54442821e-01 -4.72395016e-01
  1.33981463e+00  3.07656821e-01  1.70528474e+00 -4.75343946e-02
  1.32839193e+00 -6.06717109e-01  5.78615715e-01  7.76352029e-01
  3.13614090e-01 -8.32448659e-01  4.55719111e-01 -2.54705591e+00
 -5.02197572e-01 -1.61409775e-01  8.90623459e-02 -6.80966348e-02
  8.51593680e-02 -1.24839537e+00 -6.73210416e-01  8.43213259e-04
  2.45793692e+00  2.33285859e-01  3.65162918e-01 -1.18048979e+00
 -8.19188059e-01 -9.57642162e-01  1.40313872e+00 -1.14454082e+00
 -2.38143682e-01 -9.36722381e-01  6.19666424e-02 -1.96253210e-01
  5.51297634e-01  3.43800398e-01  1.98011275e-01 -1.62896848e+00
  7.30597186e-01 -2.51353767e-01  9.30790158e-01 -5.38900306e-02
  9.24722608e-01 -1.00467352e+00 -2.55333847e-01 -8.72762157e-01
 -7.51658033e-01  1.03448872e-01  1.47318436e+00 -7.97167194e-01
  2.07483584e-02 -8.98465150e-01  1.40342725e+00 -5.15108050e-01
  7.33990043e-01 -1.68901454e+00  2.22518236e-02  3.49946276e-01
  1.89494431e+00 -3.71727884e-01  5.30017180e-01  1.25453660e-01
  6.77890584e-01  7.16526710e-01  1.80712847e+00  3.15176040e-01
  9.65978150e-01 -1.26305534e+00  2.66475099e-01  2.79961954e-01
 -5.15038529e-01 -1.91065745e+00 -1.99278398e-01 -1.12641906e+00
 -7.84716052e-01 -7.36891901e-01  1.48375883e+00  9.33929141e-01
  1.78496688e+00 -8.33803549e-01  1.42101240e+00  8.79804253e-01
  2.82710803e+00 -9.51237489e-01 -4.96683060e-01 -2.55818844e+00
 -9.13408515e-01 -1.44973966e+00  1.83702550e+00  3.24024602e-01
  1.28986323e+00  2.00450335e-02 -5.14145726e-01 -1.32427131e+00
  8.87393123e-01  1.53956102e-01  1.53593039e+00 -1.27478934e-01
  1.56809516e+00  4.30337589e-01  1.38856178e+00 -2.90042520e-01
 -1.10455070e+00 -2.72894894e+00 -1.20501126e-02  1.32289634e-02
  1.55287194e+00  1.00162741e+00  6.21191223e-01 -1.60767136e+00
  6.78694339e-01 -3.48657409e-01  7.07219566e-01 -9.50745938e-01
  6.96192694e-01 -5.89701765e-01  9.08327929e-03 -1.36320889e-01
  1.40467688e+00  1.06803596e+00  7.14172020e-01 -1.44959998e+00
 -4.46084337e-01 -7.71298398e-01  7.39535931e-01 -1.01940558e+00
 -2.36712646e-01  2.49907138e-01  8.18116301e-01 -7.79325834e-01
 -9.37273263e-02 -7.15478551e-01 -3.85060906e-01 -7.28065485e-01
 -8.60859102e-03 -1.23344651e+00  1.15437201e-01 -1.41498030e+00
 -2.09022106e-01 -7.53797816e-01  1.48142352e-02 -8.74431327e-01
  3.22421481e-01  1.53705172e+00  1.91853674e+00 -9.19363792e-01
  4.16615838e-01  8.37510180e-01  2.02239569e+00 -2.65839535e-01
  1.14192166e+00 -3.56865097e-01  1.24387775e+00  9.04545434e-01
  3.28891867e+00  2.11145554e+00  1.12246801e+00 -7.69739060e-01
 -3.48613673e-01 -3.45535262e-01  4.20525139e-01 -1.64328504e+00
 -3.03272858e-01 -9.40416300e-01  6.55020872e-02 -9.01586808e-02
  1.31998546e+00 -5.80269306e-01  5.68624573e-01  3.90402826e-01
  1.41375163e+00  1.24712700e+00  1.68749370e+00  1.70768369e-02
 -2.40339874e-03 -2.00335818e+00 -1.30906324e+00 -1.08323996e+00] y el error es: 0.0009596379282583615
```

```
matrices = (f"/content/matTridiag{mat}.npy" for mat in files)
vectores = (f"/content/vecb{vec}.npy" for vec in files)

for mat, vec in zip(matrices, vectores):
    A = np.load(mat)
    b = np.load(vec)
    x0 = np.random.rand(b.size)
    x, N, E = gauss_seidel(A, b, x0, 100, np.finfo(float).eps**(2/3))
    print(f"La sol es: {x} y el error es: {E}")
```

```
 -1.28405518e+00 -1.54315587e+00  1.10652595e+00 -1.22073462e-02
  7.50927452e-01 -7.06384065e-01  4.39331180e-02  5.12079240e-01
  8.01219303e-01 -4.71651642e-02  6.54442794e-01 -4.72395071e-01
  1.33981459e+00  3.07656801e-01  1.70528472e+00 -4.75344062e-02
  1.32839189e+00 -6.06717134e-01  5.78615706e-01  7.76352021e-01
  3.13614030e-01 -8.32448993e-01  4.55717961e-01 -2.54705731e+00
 -5.02198588e-01 -1.61410089e-01  8.90598152e-02 -6.80977373e-02
  8.51591235e-02 -1.24839553e+00 -6.73210464e-01  8.43164883e-04
  2.45793369e+00  2.33281170e-01  3.65155864e-01 -1.18049425e+00
 -8.19192301e-01 -9.57645210e-01  1.40313554e+00 -1.14454443e+00
 -2.38145609e-01 -9.36723895e-01  6.19631608e-02 -1.96265106e-01
  5.51282780e-01  3.43773996e-01  1.97974659e-01 -1.62899604e+00
  7.30587755e-01 -2.51358224e-01  9.30788673e-01 -5.38906461e-02
  9.24722293e-01 -1.00467357e+00 -2.55333868e-01 -8.72762190e-01
 -7.51658060e-01  1.03448814e-01  1.47318390e+00 -7.97167845e-01
  2.07473526e-02 -8.98465626e-01  1.40342703e+00 -5.15108111e-01
  7.33990023e-01 -1.68901454e+00  2.22518298e-02  3.49946279e-01
  1.89494431e+00 -3.71727911e-01  5.30017146e-01  1.25453578e-01
  6.77890509e-01  7.16522707e-01  1.80712067e+00  3.15165338e-01
  9.65971578e-01 -1.26305686e+00  2.66477415e-01  2.79966597e-01
 -5.15015882e-01 -1.91063440e+00 -1.99256315e-01 -1.12640070e+00
 -7.84699690e-01 -7.36886996e-01  1.48375589e+00  9.33922451e-01
  1.78496107e+00 -8.33813956e-01  1.42100808e+00  8.79801724e-01
  2.82710711e+00 -9.51238019e-01 -4.96683927e-01 -2.55819012e+00
 -9.13409649e-01 -1.44974227e+00  1.83702532e+00  3.24031660e-01
  1.28987116e+00  2.00545979e-02 -5.14137732e-01 -1.32426464e+00
  8.87394596e-01  1.53955349e-01  1.53592934e+00 -1.27482875e-01
  1.56808752e+00  4.30334136e-01  1.38856040e+00 -2.90042879e-01
 -1.10455070e+00 -2.72894893e+00 -1.20501078e-02  1.32289637e-02
  1.55287194e+00  1.00162740e+00  6.21191075e-01 -1.60767093e+00
  6.78695265e-01 -3.48656491e-01  7.07219639e-01 -9.50746254e-01
  6.96191100e-01 -5.89703926e-01  9.08241427e-03 -1.36321245e-01
  1.40467660e+00  1.06803538e+00  7.14166646e-01 -1.44960648e+00
 -4.46093494e-01 -7.71320309e-01  7.39506753e-01 -1.01946227e+00
 -2.36791708e-01  2.49814361e-01  8.18064442e-01 -7.79346154e-01
 -9.37369222e-02 -7.15479810e-01 -3.85061301e-01 -7.28065632e-01
 -8.60873505e-03 -1.23344686e+00  1.15435294e-01 -1.41498357e+00
 -2.09024600e-01 -7.53798699e-01  1.48138855e-02 -8.74431314e-01
  3.22421933e-01  1.53705278e+00  1.91853718e+00 -9.19363234e-01
  4.16616080e-01  8.37510277e-01  2.02239571e+00 -2.65839530e-01
  1.14192749e+00 -3.56851511e-01  1.24388644e+00  9.04551096e-01
  3.28891985e+00  2.11145536e+00  1.12246752e+00 -7.69739654e-01
 -3.48614505e-01 -3.45538183e-01  4.20510568e-01 -1.64330468e+00
 -3.03281642e-01 -9.40417772e-01  6.54959499e-02 -9.01792485e-02
  1.31994004e+00 -5.80331566e-01  5.68516501e-01  3.90303984e-01
  1.41362827e+00  1.24694897e+00  1.68727487e+00  1.69821846e-02
 -2.49333947e-03 -2.00340740e+00 -1.30908781e+00 -1.08324274e+00] y el error es: 3.428283832123252e-11
```

```
matrices = (f"/content/matTridiag{mat}.npy" for mat in files)
vectores = (f"/content/vecb{vec}.npy" for vec in files)

for mat, vec in zip(matrices, vectores):
    A = np.load(mat)
    b = np.load(vec)
    x0 = np.random.rand(b.size)
    x, N, E = gauss_seidel(A, b, x0, 1000, np.finfo(float).eps**(2/3))
    print(f"La sol es: {x} y el error es: {E}")
```

```
La sol es: [0.6535329  0.5315622  1.24682267 1.13584452 1.46384978] y el error es: 1.7549959034320085e-11
La sol es: [nan nan nan nan nan nan nan nan nan nan] y el error es: nan
<ipython-input-2-4e5ccb5c055f>:9: RuntimeWarning: overflow encountered in matmul
  if (E:=norm(A @ x - b)) < t:
<ipython-input-2-4e5ccb5c055f>:14: RuntimeWarning: overflow encountered in double_scalars
  x[i] = (b[i]-A[i, :i]@x[:i] - A[i, i+1:]@x0[i+1:])/A[i, i]
<ipython-input-2-4e5ccb5c055f>:14: RuntimeWarning: overflow encountered in matmul
  x[i] = (b[i]-A[i, :i]@x[:i] - A[i, i+1:]@x0[i+1:])/A[i, i]
<ipython-input-2-4e5ccb5c055f>:14: RuntimeWarning: invalid value encountered in matmul
  x[i] = (b[i]-A[i, :i]@x[:i] - A[i, i+1:]@x0[i+1:])/A[i, i]
<ipython-input-2-4e5ccb5c055f>:9: RuntimeWarning: invalid value encountered in matmul
  if (E:=norm(A @ x - b)) < t:
La sol es: [-1.58881138e-01 -5.63844604e-01  9.49679122e-01  4.31619788e-01
  1.08844912e+00 -1.37572293e+00 -2.57413194e-01 -1.03268116e+00
  7.14070861e-01  3.30701298e-01  8.83156737e-01 -1.48885351e+00
  1.07086616e+00  1.27007593e-01  2.18520754e+00  9.76404562e-01
  1.34960537e+00 -6.41972141e-01  3.73030966e-01  6.02734735e-01
  1.19669259e+00 -4.02074123e-01  1.06872931e+00 -1.33620999e+00
 -4.38350229e-01  2.65463055e-02  3.25288063e+00  3.10333860e+00
  1.37956977e+00 -1.02737128e+00 -2.00050114e-01 -7.98425758e-01
  9.72281275e-01  2.91787590e-01  6.75469344e-01 -7.53582552e-01
  3.25960111e-01 -9.80218623e-01  4.21674061e-01 -2.22779514e-01
  3.71943915e-01 -1.04447299e+00  5.21751873e-01 -5.38953174e-01
  1.31196145e+00  4.69579571e-01  1.58504795e+00 -1.29055680e+00
```

```
 1.60721818e-01 -1.38321799e+00  2.14768207e+00 -7.24615115e-01
-3.47211566e-01 -3.35494889e-01  1.01755499e+00  8.55818729e-02
 9.91095968e-01 -2.40147596e-01  8.32984648e-01 -9.09657996e-01
-5.35497325e-01 -6.76609069e-01  1.26680532e+00 -9.27773838e-01
 9.12382058e-01 -3.68519095e-01  2.11962672e-01 -1.23761204e+00
 4.83663421e-01  3.07157633e-01  6.03837381e-01 -1.40192083e+00
-6.38724321e-01 -1.78485569e+00 -7.99674432e-01 -1.71602670e+00
-6.09642115e-01 -9.13228580e-01  3.50876374e-01 -1.37121850e+00
 7.02959692e-01 -5.85979870e-01  1.10801862e+00 -8.82787401e-01
 1.56630078e-01 -2.48950600e-01  1.83601417e+00  2.20081119e-01
 3.41397764e-01 -1.22804428e+00  9.12573202e-01 -1.27082245e+00
 4.75389451e-01 -4.91383050e-01  2.10707053e-01 -1.62218294e+00
-1.03424576e+00 -3.03552449e-01  8.56523560e-01 -6.69503826e-02
 1.00791441e+00 -3.92130210e-01 -2.60003785e-01 -6.30413420e-01
 7.60127397e-01 -6.00990136e-01  4.45747641e-01 -1.80941224e+00
-9.38260323e-01  1.63625925e-01  1.06117758e+00 -3.38634050e-01
 5.77349390e-01 -1.49417056e+00 -3.33297002e-01 -2.30741932e+00
-1.49145527e+00 -1.31365639e+00  5.50154551e-01 -1.42678163e+00
-2.87952830e-01 -1.73784186e-02  8.88290559e-01  7.21561547e-01
 1.30160114e+00 -1.27144294e+00 -1.09251679e+00 -8.34813164e-01
 9.58076715e-01  5.39079673e-01  6.89518903e-01 -1.52992068e+00
 2.74664740e-02 -8.57729523e-01 -1.74179459e-02 -2.35878882e-01
-1.60283844e-01 -1.94709202e+00 -4.96058308e-01 -2.20934807e-01
 5.53423605e-01 -1.56747875e-01  4.28246557e-03 -1.72422061e+00
-1.81259715e-01 -1.42006737e+00  3.56436243e-01 -6.68008627e-01
 1.10458340e+00 -1.28270928e+00  8.66753735e-01  7.70395161e-01
 1.15373024e+00 -5.11998997e-01  4.49513359e-01 -7.81811333e-01
-1.67269569e+00 -3.26999917e+00 -8.90139777e-01 -1.41275929e+00
-4.48600522e-01 -2.98956041e+00 -2.62771980e+00 -1.47064342e+00
 1.81655864e+00  7.30571927e-01 -1.13788601e-01 -1.57287318e+00
-7.26510209e-01 -1.47183413e+00  1.44614556e+00  6.66599411e-01
 2.61561314e-02 -3.25472206e-01  3.02487736e-01 -7.23315201e-01
 7.50611173e-01  1.83873927e-01  3.58991531e-01 -1.31171923e+00
-1.27192874e+00 -1.27824542e+00  8.41674017e-01  1.04784661e-01
```

Podemos ver que el metodo general tiene menor error que el metodo mas especifico.

Para el caso de la matriz de 5 y 500 elementos, con pocas iteraciones se tiene una buena aproximación y mientras mas iteraciones se reduce el error. Por otro lado la matriz de 10 elementos mientras mas iteraciones el error crece mas y mas.

_

Ejercicio 2 (6 puntos)

Programar el algoritmo de factorización de Cholesky y resuelva un sistema de ecuaciones lineales $\mathbf{A}\mathbf{x} = \mathbf{b}$, donde la matriz $\mathbf{A}$ es simétrica.

1. Escriba la función `factChol` calcula la matriz $\mathbf{L}$ de la factorización de Cholesky. Esta función recibe como parámetros:

- una matriz $\mathbf{A}$,
- una tolerancia τ.

  Dentro de la función se reserva memoria para la matriz $\mathbf{L}$ y se inicializa como la matriz cero. Si al calcular l_{jj} el radicando de la expresión es negativo o $|l_{jj}|$ es menor que la tolerancia τ, devolver `None`. Si no ocurre lo anterior, devuelve $\mathbf{L}$.

2. Escriba la función que resuelve el sistema $\mathbf{A}\mathbf{x} = \mathbf{b}$ cuando $\mathbf{A}$ es una simétrica. La función debe recibir como parámetros:

- la matriz $\mathbf{A}$,
- el vector $\mathbf{b}$,
- una tolerancia τ.

  La salida de la función es $\mathbf{x}$ si la matriz $\mathbf{A}$ es definida positiva o `None` si no se logró factorizar la matriz.

Use la funci'on del `factChol` primer punto para calcular la factorización de Cholesky de $\mathbf{A}$. Si el resultado es el arreglo vacío, terminar devolviendo `None`. En caso contrario, use la función `numpy.transpose` o `L.T` para obtener la matriz transpuesta $\mathbf{L}^\top$. Resuelva los sistemas de ecuaciones para la matrices trianguales.\ Devolver $\mathbf{x}$ o `None` según sea el caso.

3. Pruebe la función anterior usando los datos en el archivo `datosTarea06.zip`

Matriz:	Vector:
matA005.npy	vecb005
matA010.npy	vecb010
matA050.npy	vecb050
matA500.npy	vecb500

- Imprima el tamaño de la matriz.
- Reporte si el sistema tuvo o no solución. En caso de que sí fue obtenida la solución, imprima las primeras y últimas tres entradas del vector solución.
- Imprima el valor del error $\|\mathbf{Ax} - \mathbf{b}\|$.

Solución:

```
## Funciones de la tarea 4

def forwardSubstitution(L: np.ndarray, b: np.ndarray, t):
  d = L.shape[0]
  res = np.zeros(d)
  for i in range(d):
    sum = np.dot(res[:i], L[i, :i])
    if np.abs(L[i, i]) ⩽ t:
        print(L[i, i])
        return None
    x = (b[i] - sum)/L[i, i]
    res[i] = x

  return res

def backwardSubstitution(U: np.ndarray, b: np.ndarray, t):
    d = U.shape[0]
    res = np.zeros(d)
    for i in reversed(range(d)):
        sum = np.dot(res[i:], U[i, i:])
        if abs(U[i, i]) < t:
            print(U[i, i], t)
            return None
        x = (b[i] - sum)/U[i, i]
        res[i] = x
    return res

```

```
import numpy as np

def factchol(A: np.ndarray, t:float):
    L = np.zeros(A.size).reshape(A.shape)
    for i in range(A.shape[0]):
        if (r := A[i, i] - sum(L[i, :i]**2)) < 0 or abs(np.sqrt(r)) < t:
            return None
        L[i, i] = np.sqrt(r)
        for j in range(i+1, A.shape[0]):
            L[j, i] = (A[j, i]- L[j,:i].dot(L[i, :i]))/L[i, i]
    return L



```

```
def SolveCholesky(A: np.ndarray, b:np.ndarray, t):
    if (L := factchol(A, t)) is not None:
        y = forwardSubstitution(L, b, t)
        x = backwardSubstitution(L.T, y, t)
        return x
```

```
files = ['005', '010', '050', '500']
matrices = (f"/content/matA{mat}.npy" for mat in files)
vectores = (f"/content/vecb{vec}.npy" for vec in files)
```

```python

for mat, vec in zip(matrices, vectores):
    A = np.load(mat)
    b = np.load(vec)
    x = SolveCholesky(A, b, np.finfo(float).eps**(2/3))
    print(x, np.linalg.norm(A @ x - b))

```

```
[1. 1. 1. 1. 1.] 1.0877919644084146e-15
[-4.13705776 -5.46565854  2.91781122 -2.43456113  4.02049825 -5.34744604
  0.38357026 -3.50959056  3.60081682 -3.25      ] 1.076401158743041e-15
[ 1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.] 3.598503956136582e-15
[ 1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.
  1. -2.  3. -1.  2. -3.  1. -2.  3. -1.  2. -3.  1. -2.] 2.2499605032669807e-14
```

```
Podemos ver que el error es muy pequeño y en los cuatro casos
obtuvimos solución.
```

Podemos ver que el error es muy pequeño y en los cuatro casos obtuvimos solución.