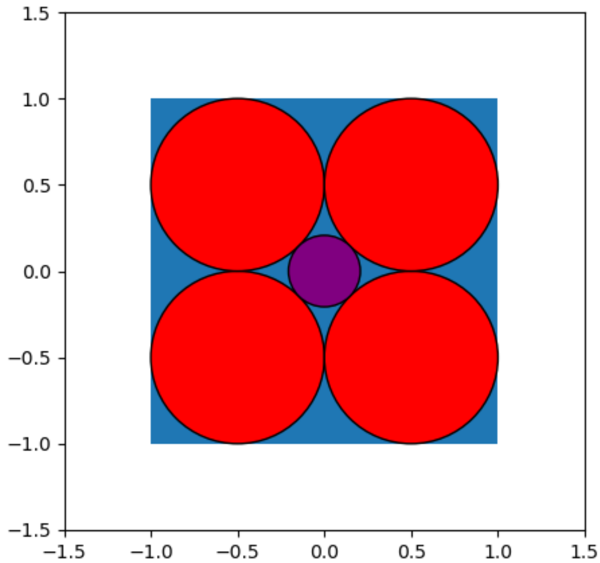
Exercise 1

(a)

Situation for l=2 and s=2:

$$Q_{2,2} = \{[x_1 x_2]^T \in \mathbb{R}^2 \mid |x_i| \leq 1 \text{ for all } i = 1, 2\} = [-1, 1]^2$$



(b)

First, let's calculate the radius of the inner hyperball for any $l \in \mathbb{N}, s \in \mathbb{R}_{>0}$:
The distance from the center of the inner hyperball (equal to the center of the hypercube) to the center of one of the 2^l outter hyperballs (doesn't matter which one) can be calculated the following:

$$d \coloneqq \sqrt{l \cdot \left(\frac{s}{4}\right)^2}$$

Thus, the radius of the inner hyperball is equal to:

$$r \coloneqq d - \frac{s}{4} = \sqrt{l \cdot \left(\frac{s}{4}\right)^2} - \frac{s}{4}$$

Now, we simply must solve the following inequality to find an $l \in \mathbb{N}$ for an arbitrary but fixed $s \in \mathbb{R}_{>0}$ such that $B(Q_{l,s}) \nsubseteq Q_{l,s}$:

$$\begin{aligned}
\frac{s}{2} &< r \\
\frac{s}{2} &< d - \frac{s}{4} \\
\frac{s}{2} &< \sqrt{l \cdot \left(\frac{s}{4}\right)^2} - \frac{s}{4} \\
\frac{3 \cdot s}{4} &< \sqrt{l} \cdot \frac{s}{4} \\
3 &< \sqrt{l} \\
l &> 9
\end{aligned}$$

Exercise 2

(a)

Let's first find all eigenvalues of A_c:

$$\begin{aligned}
&\{\lambda \in \mathbb{R} \mid \det\left(A_c - \lambda \cdot I_3\right) = 0\} \\
&\{\lambda \in \mathbb{R} \mid \det \begin{bmatrix} 2-\lambda & 0 & c \\ 0 & 1-\lambda & 0 \\ c & 0 & 1-\lambda \end{bmatrix} = 0\} \\
&\{\lambda \in \mathbb{R} \mid -\lambda^3 + 4 \cdot \lambda^2 + c^2 \cdot \lambda - 5 \cdot \lambda + 2 - c^2\} \\
&\{\lambda \in \mathbb{R} \mid (\lambda - 1) \cdot (-\lambda^2 + 3\dot{\lambda} + c^2 - 2)\}
\end{aligned}$$

$$\{1, \frac{3 - \sqrt{4 \cdot c^2 + 1}}{2}, \frac{3 + \sqrt{4 \cdot c^2 + 1}}{2}\} =: \Lambda_{A_c}$$

So, if for all $\lambda \in \Lambda_{A_c}$ it must hold that $\lambda \geq 0$, we must set c such that the following inequality holds:

$$\begin{aligned}
\frac{3 - \sqrt{4 \cdot c^2 + 1}}{2} &\geq 0 \\
3 - \sqrt{4 \cdot c^2 + 1} &\geq 0 \\
3 &\geq \sqrt{4 \cdot c^2 + 1} \\
9 &\geq 4 \cdot c^2 + 1 \\
8 &\geq 4 \cdot c^2 \\
2 &\geq c^2 \\
c &\in (-\sqrt{2}, \sqrt{2}) \subseteq \mathbb{R}
\end{aligned}$$

(b)

Using Theorem 5.19, we create an orthogonal matrix $U \in \mathbb{R}^{n \times n}$ with the columns being the eigenvectors of A. We can now use Theorem 5.20 to create the diagonal matrix $\Lambda \in \mathbb{R}^{n \times n}$ where the eigenvalue-entry $\lambda_{i,i}$ corresponds to eigenvector in the column i of U.
A simple verification:

$$\begin{aligned}
A &= U \Lambda U^\top \\
A &= U \Lambda U^{-1} \\
AU &= U \Lambda \\
Av_i &= v_i \lambda_{i,i} && , \forall i \in [n], v_i := col_i(U) \\
Av_i &= \lambda_{i,i} v_i && , \forall i \in [n], v_i := col_i(U)
\end{aligned}$$

Since A is positive semi-definite, all entries $\lambda_{i,i} \geq 0$. Thus, we can create a diagonal matrix Λ' consisting of the entries $\lambda'_{i,i} \coloneqq \sqrt{\lambda_{i,i}}$. Thus, $\Lambda = \Lambda'\Lambda'^\top$. Let $B \coloneqq UU\Lambda'$. Now:

$$\begin{aligned}
A &= U\Lambda U^\top \\
A &= U(\Lambda'\Lambda'^\top)U^\top \\
A &= U(\Lambda'\Lambda'^\top)U^\top \\
A &= (U\Lambda')(\Lambda'^\top U^\top) \\
A &= (U\Lambda')(U\Lambda')^\top \\
A &= BB^\top
\end{aligned}$$

(c)

Since A is symmetric, the following holds $\forall x, y \in R^n$:

$$\langle Ax, y\rangle = \langle x, Ay\rangle$$

Let $v_1 \in E_1, v_2 \in E_2$. Then $Av_1 = \lambda_1 v_1$ and $Av_2 = \lambda_2 v_2$.

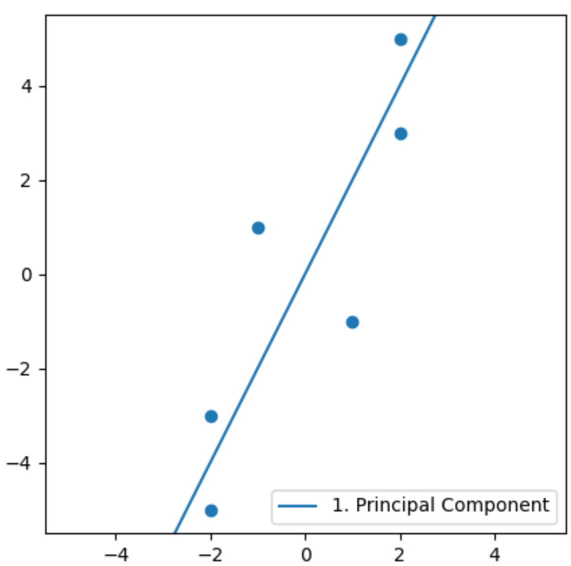
$$\begin{aligned}
0 &= \langle Av_1, v_2\rangle - \langle v_1, Av_2\rangle \\
&= \langle \lambda_1 v_1, v_2\rangle - \langle v_1, \lambda_2 v_2\rangle \\
&= (\lambda_1 - \lambda_2)\langle v_1, v_2\rangle
\end{aligned}$$

Since $\lambda_1 \neq \lambda_2$ it follows, that $\langle v_1, v_2\rangle = 0$.

Exercise 3

(a)

Plot and estimate first principal component $(\frac{1}{\sqrt{5}}, \frac{2}{\sqrt{5}})$



(b)

$$\begin{aligned} C &\coloneqq A^\top A \\ &= \begin{bmatrix} -2 & -2 & -1 & 1 & 2 & 2 \\ -5 & -3 & 1 & 1 & 3 & 5 \end{bmatrix} \begin{bmatrix} -2 & -5 \\ -2 & 3 \\ -1 & 1 \\ 1 & -1 \\ 2 & 3 \\ 2 & 5 \end{bmatrix} \\ &= \begin{bmatrix} 18 & 18 \\ 18 & 70 \end{bmatrix} \end{aligned}$$

(c)

View code in the appendix for the calculations.
The eigenvector we got is: $\begin{pmatrix} 0.298 \\ 0.954 \end{pmatrix}$

(d)

Well, they both point in the same general direction with a slope greater than 1. Our slope came out to be at 2, since eye-balling stuff tends to round to integer values. The more optimal slope however seems to be even greater than $3 < \frac{0.954}{0.298}$.

Exercise 4

(a)

$$S \coloneqq \begin{bmatrix} 0&0&1&1&1&0&0&0 \\ 0&0&1&1&0&0&0&0 \\ 1&1&0&1&0&0&0&0 \\ 1&1&1&0&0&0&0&1 \\ 1&0&0&0&0&1&1&1 \\ 0&0&0&0&1&0&1&1 \\ 0&0&0&0&1&1&0&0 \\ 0&0&0&1&1&1&0&0 \end{bmatrix}$$

$$D \coloneqq \begin{bmatrix} 3&0&0&0&0&0&0&0 \\ 0&2&0&0&0&0&0&0 \\ 0&0&3&0&0&0&0&0 \\ 0&0&0&4&0&0&0&0 \\ 0&0&0&0&4&0&0&0 \\ 0&0&0&0&0&3&0&0 \\ 0&0&0&0&0&0&2&0 \\ 0&0&0&0&0&0&0&3 \end{bmatrix}$$

$$L \coloneqq D - S = \begin{bmatrix} 3 & 0 & -1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 2 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 4 & 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 & 4 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 3 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 & -1 & 2 & 0 \\ 0 & 0 & 0 & -1 & -1 & -1 & 0 & 3 \end{bmatrix}$$

(b)

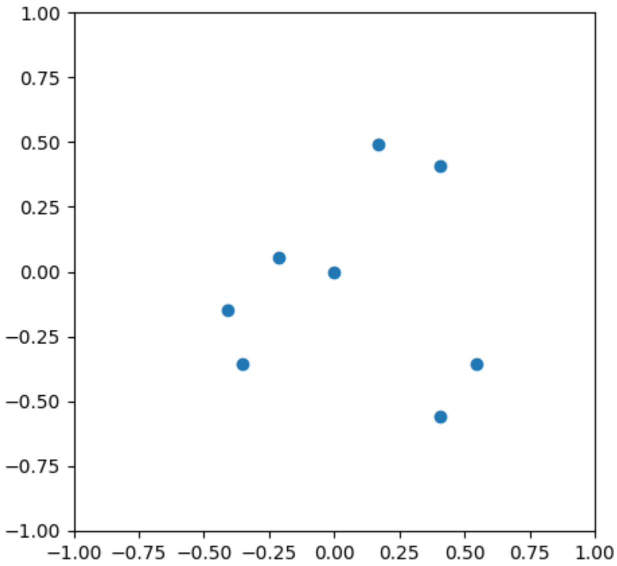
View code in the appendix for the calculations.

$$\lambda_1 = 0.0$$
$$u_1 = \begin{pmatrix} -0.354 \\ 0.169 \\ -0.408 \\ 0.408 \\ 0.548 \\ -0.214 \\ 0.408 \\ -0.002 \end{pmatrix}$$
$$\lambda_2 = 0.657$$
$$u_2 = \begin{pmatrix} -0.354 \\ 0.493 \\ -0.149 \\ -0.558 \\ -0.358 \\ 0.056 \\ 0.408 \\ -0.002 \end{pmatrix}$$

(c)

The eigenvalue λ_1 is extremely close to 0. Therefore, $\lambda_1 u_1$ is very close to 0, so also Au_1 must be very close to 0.

(d)



(e)

The Spectral Clustering Algorithm returns the clusters computed using the k-means clustering algorithm on the points $((u_1)_i, (u_2)_i), i = 1, \dots, 8$. Therefore, depending on the starting centroids, we might see a cluster containing only the two bottom right points, and another containing the remaining points.

Exercise 5

Appendix

Code for Exercise 3

```python
import functools
import math


def length(v):
    return math.sqrt(functools.reduce(lambda a, b: a + b, map(lambda v_i:
   v_i ** 2, v)))


def truncate(number, decimals=0):
    """
    Returns a value truncated to a specific number of decimal places.
    """
    if not isinstance(decimals, int):
        raise TypeError("decimal places must be an integer.")
    elif decimals < 0:
        raise ValueError("decimal places has to be 0 or more.")
    elif decimals == 0:
        return math.trunc(number)
```

```python

    factor = 10.0 ** decimals
    return math.trunc(number * factor) / factor


def matrix_times_vector(A, v):
    Av = list()
    for i in range(len(A)):
        assert(len(A[i]) == len(v))
        Av.append(functools.reduce(
            lambda a, b: a + b, map(lambda item: A[i][item[0]] * item[1],
    enumerate(v))))
    return Av


def power_iteration(A, x):
    x_length = length(x)
    v = tuple(map(lambda x_i: x_i / x_length, x))
    old_trunc_v = tuple(map(lambda v_i: truncate(v_i, 3), v))
    while True:

        Av = matrix_times_vector(A, v)
        Av_length = length(Av)
        v = tuple(map(lambda v_i: v_i / Av_length, Av))

        # check if does not converge no more
        new_trunc_v = tuple(map(lambda v_i: truncate(v_i, 3), v))
        if(old_trunc_v == new_trunc_v):
            break
        old_trunc_v = new_trunc_v
    return v


if __name__ == '__main__':
    C = [
        [18, 18],
        [18, 70]
    ]
    v = (1,1)
    v_power = power_iteration(C,v)
    print(tuple(map(lambda v_i: round(v_i, 3), v_power)))
```

Code for Exercise 4

```python
import numpy as np
from numpy import linalg as LA

L = np.array([
    [3, 0, -1, -1, -1, 0, 0, 0],
    [0, 2, -1, -1, 0, 0, 0, 0],
    [-1, -1, 3, -1, 0, 0, 0, 0],
    [-1, -1, -1, 4, 0, 0, 0, -1],
    [-1, 0, 0, 0, 4, -1, -1, -1],
    [0, 0, 0, 0, -1, 3, -1, -1],
    [0, 0, 0, 0, -1, -1, 2, 0],
    [0, 0, 0, -1, -1, -1, 0, 3]
])

eigenvalues, eigenvectors = LA.eig(L)

```

```
eigen_pairs = [(eigenvalues[i], eigenvectors[i]) for i in range(len(
    eigenvalues))]
eigen_pairs.sort(key=lambda tuple: tuple[0])

eigen_pairs = list(map(lambda tuple: (round(tuple[0], 3), np.round(tuple[1],
     3)), eigen_pairs))

print(eigen_pairs[0])
print(eigen_pairs[1])
```