Rules:

- Unless otherwise stated, all answers must be mathematically justified.
- Partial answers will be graded.
- Your submission has to be uploaded to Gradescope. In Gradescope, indicate the page on which each problem is written.
- You can work in groups but each student must write his/her/their own solution based on his/her/their own understanding of the problem. Please list on your submission the students you work with for the homework (this will not affect your grade).
- Problems with a ($\star$) are extra credit, they will not (directly) contribute to your score of this homework. However, for every 4 extra credit questions successfully answered your lowest homework score get replaced by a perfect score.
- If you have any questions, feel free to ask them on Ed Discussion (so that everyone can benefit from the answer) or stop at the office hours.

Problem 5.1 (2 points)**.** *Give an orthonormal basis of* $\mathbb{R}^3$ *using the Gram-Schmidt algorithm starting from the linearly independent family* (v_1, v_2, v_3) *where* $v_1 = (2, 1, -2)$*,* $v_2 = (0, 3, 3)$ *and* $v_3 = (0, 0, 1)$*.*

We could start from $\vec{v}_3$ and normalize it as our first orthonormal basis vector and we get$\vec{q}_1 = \frac{\vec{v}_3}{\|\vec{v}_3\|_2} = (0, 0, 1)$
We then choose $\vec{v}_2$ to apply the process $\vec{z}_2 = \vec{v}_2 - \langle \vec{v}_2, \vec{q}_1 \rangle \vec{q}_1 = (0, 3, 3) - 3(0, 0, 1) = (0, 3, 0)$, $\vec{q}_2 = \frac{\vec{z}_2}{\|\vec{z}_2\|_2} = (0, 1, 0)$.
Finally the $\vec{v}_1$, we get $\vec{z}_3 = \vec{v}_1 - \langle \vec{v}_1, \vec{q}_1 \rangle \vec{q}_1 - \langle \vec{v}_1, \vec{q}_2 \rangle \vec{q}_2 = (2, 1, -2) - (-2) \times (0, 0, 1) - 1 \times (0, 1, 0) = (2, 0, 0)$, $\vec{q}_3 = \frac{\vec{z}_3}{\|\vec{z}_3\|_2} = (1, 0, 0)$.
Thus the orthonormal basis that we get is $\{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$.

Problem 5.2 (2 points). *Consider the three subspaces* $U = \mathrm{Span}((1,0))$, $V = \mathrm{Span}((0,1))$, *and* $W = \mathrm{Span}((1,1))$ *of* $\mathbb{R}^2$.

(**a**) *Compute the canonical matrices* $M_U, M_V, M_W \in \mathbb{R}^{2\times 2}$ *of the orthogonal projections* P_U, P_V, P_W *onto the subspaces* U, V, W.

$M_U = \frac{\vec{u}\vec{u}^\top}{\langle \vec{u},\vec{u}\rangle} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$, $M_V = \frac{\vec{v}\vec{v}^\top}{\langle \vec{v},\vec{v}\rangle} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$, $M_W = \frac{\vec{w}\vec{w}^\top}{\langle \vec{w},\vec{w}\rangle} = \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix}$

(**b**) *Do we have* $M_U M_V = M_V M_U$? *Draw the situation.*

$M_U M_V = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$, $M_V M_U = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$, *thus* $M_U M_V = M_V M_U$.

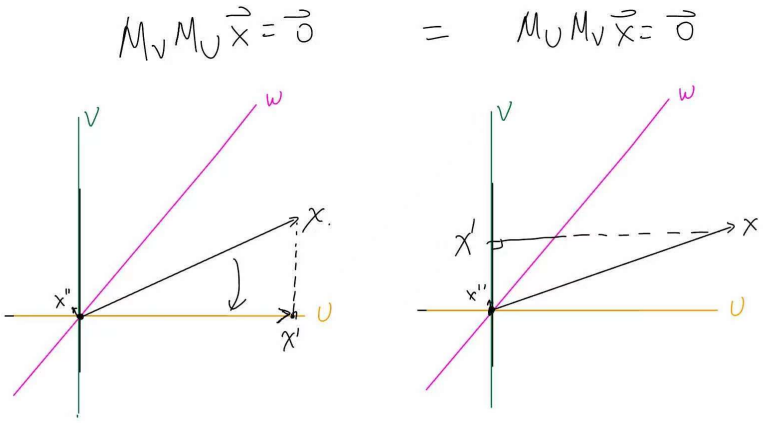


Figure 1: Equal Case

(**c**) *Do we have* $M_U M_W = M_W M_U$? *Draw the situation.*

$M_U M_W = \begin{bmatrix} 1/2 & 1/2 \\ 0 & 0 \end{bmatrix}$, $M_W M_U = \begin{bmatrix} 1/2 & 0 \\ 1/2 & 0 \end{bmatrix}$, *thus* $M_U M_W \neq M_W M_U$.

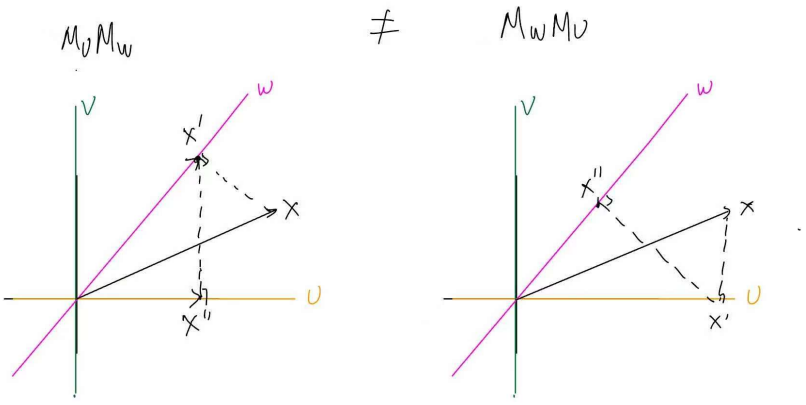


Figure 2: Non-Equal Case

Problem 5.3 (3 points). *Let $\mathcal{B} = (u_1, \ldots, u_n)$ be an orthonormal basis of $\mathbb{R}^n$ and*

$$U = \begin{pmatrix} | & & | \\ u_1 & \cdots & u_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{n \times n}.$$

A vector $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ can thus also be represented by its coordinates $(\langle x, u_1 \rangle, \ldots, \langle x, u_n \rangle) \in \mathbb{R}^n$ in the basis $\mathcal{B}$. Consider L a linear transformation from $\mathbb{R}^n$ to $\mathbb{R}^n$ and denote by $\tilde{L} \in \mathbb{R}^{n \times n}$ its canonical matrix.

(**a**) *Let $\tilde{L}' \in \mathbb{R}^{n \times n}$ be the matrix of L in the basis $\mathcal{B}$, i.e., the matrix which computes the coordinates of Lx in the basis $\mathcal{B}$ from the coordinates of x in the basis $\mathcal{B}$. Express $\tilde{L}'$ in terms of $\tilde{L}$ and U.*

Suppose the linear transformation matrix from canonical basis to canonical basis is $\widetilde{L}$, then for $\forall \vec{x}, \vec{y} \in \mathbb{R}^n$ under canonical basis we have

$$\vec{y} = \widetilde{L}(\vec{x}) \tag{1}$$

, where $\vec{x}, \vec{y}$ is the coordinate under canonical basis. Now given any $\vec{x}$ and $\vec{y}$ under canonical basis, we have $\vec{y} = U\vec{y}^*$ and $\vec{x} = U\vec{x}^*$ where $\vec{x}^*$ and $\vec{y}^*$ are coordinates under basis $\mathcal{B}$. Plug into (1) we could get $U\vec{y}^* = \widetilde{L}(U\vec{x}^*)$. By the property of linear transformation, $U\vec{y}^* = \widetilde{L}(U\vec{x}^*) = \widetilde{L}U\vec{x}^*$. Multiply on both sides by $U^\top$ we get $\vec{y}^* = U^\top \widetilde{L} U \vec{x}^*$. Thus we could write $\widetilde{L}' = U^\top \widetilde{L} U$.
A more rigorous way on showing this is from definition:
Consider any coordinate $\vec{x}$ under canonical basis, we have $\vec{x} = U\vec{x}^*$, where $\vec{x}^*$ is the coordinate under basis $\mathcal{B}$. Then $\forall \vec{u}_i$ under canonical basis, we have $L\vec{u}_i = \widetilde{L}\vec{u}_i$ by definition. Meanwhile, $L\vec{u}_i = U(L\vec{u}_i)^*$ (where $(L\vec{u}_i)^*$ is the coordinate under basis $\mathcal{B}$). So $(L\vec{u}_i)^* = U^\top L\vec{u}_i$. Finally we apply such process for all $\vec{u}_i, \forall i = 1, 2, \cdots, n$, we have $\widetilde{L}' = U^\top \widetilde{L} U$.

(**b**) *Show that $\tilde{L}'$ is symmetric if and only if $\tilde{L}$ is (symmetry can be tested in any orthogonal basis).*

Using the property that $UU^\top = U^\top U = I_n$, we could go bidirectionally:

(a) $\Longrightarrow$ We want to examine whether $\widetilde{L}^\top = \widetilde{L}$. Since $\widetilde{L}'$ is symmetric, we know $\widetilde{L}'^\top = \widetilde{L}'$ and thus $(U^\top \widetilde{L} U)^\top = U^\top \widetilde{L} U$. We do the arithmetic operations as follows:

$$\begin{aligned} (U^\top \widetilde{L} U)^\top &= U^\top \widetilde{L} U \\ U^\top \widetilde{L}^\top U &= U^\top \widetilde{L} U && \text{(Property of matrix transpose)} \\ \widetilde{L}^\top U &= \widetilde{L} U && \text{(Multiply U on the left hand side)} \\ \widetilde{L}^\top &= \widetilde{L} && \text{(Multiply U transpose on the right hand side)} \end{aligned}$$

, which proves the forward direction.

(b) $\Longleftarrow$ We want to examine whether $\widetilde{L}'^\top = \widetilde{L}'$.

$$\begin{aligned} (U^\top \widetilde{L} U)^\top &= U^\top \widetilde{L}^\top U \\ &= U^\top \widetilde{L} U && \text{(Using assumption in this direction)} \\ &= \widetilde{L}' && \text{(By definition from part(a))} \end{aligned}$$

, which proves the backward direction.

(**c**) *Show that $\tilde{L}'$ is orthogonal if and only if $\tilde{L}$ is (orthogonality can be tested in any orthogonal basis).*

Since $\widetilde{L}'$ and $\widetilde{L}$ are both square matrix, we can examine whether $\widetilde{L}'^{\top}\widetilde{L}' = \mathbf{I}_n$ and $\widetilde{L}^{\top}\widetilde{L} = \mathbf{I}_n$. Using the property that $UU^{\top} = U^{\top}U = I_n$, we could go bidirectionally:

(a) $\Longrightarrow$ We want to examine whether $\widetilde{L}^{\top}\widetilde{L} = I_n$. Since $\widetilde{L}'$ is orthogonal, we know $\widetilde{L}'^{\top}\widetilde{L}' = I_n$ and thus $(U^{\top}\widetilde{L}U)^{\top}U^{\top}\widetilde{L}U = I_n$. We do the arithmetic operations as follows:

$$\begin{aligned}(U^{\top}\widetilde{L}U)^{\top}U^{\top}\widetilde{L}U &= U^{\top}\widetilde{L}^{\top}UU^{\top}\widetilde{L}U && \text{(Property of matrix transpose)}\\ &= U^{\top}\widetilde{L}^{\top}\widetilde{L}U = I_n && \text{(Property of matrix U)}\end{aligned}$$

, now multiply U on the left and $U^{\top}$ on the right side we get $\widetilde{L}^{\top}\widetilde{L} = UU^{\top} = I_n$, which proves the forward direction.

(b) $\Longleftarrow$ We want to examine whether $\widetilde{L}'^{\top}\widetilde{L}' = I_n$. Since $\widetilde{L}$ is orthogonal, we know $\widetilde{L}^{\top}\widetilde{L} = I_n$ and thus We do the arithmetic operations as follows:

$$\begin{aligned}(U^{\top}\widetilde{L}U)^{\top}U^{\top}\widetilde{L}U &= U^{\top}\widetilde{L}^{\top}UU^{\top}\widetilde{L}U && \text{(Property of matrix transpose)}\\ \widetilde{L}'^{\top}\widetilde{L}' &= U^{\top}\widetilde{L}^{\top}\widetilde{L}U\\ \widetilde{L}'^{\top}\widetilde{L}' &= I_n && \text{(Multiply U on the left and U transpose on the right)}\end{aligned}$$

, which proves the backward direction.

(**d**) *Let M be another linear transformation from $\mathbb{R}^n$ to $\mathbb{R}^n$. Show that $\widetilde{(LM)}'$, the matrix of LM in the basis $\mathcal{B}$, is equal to $\tilde{M}'\tilde{L}'$.*

Suppose the linear transformation matrix from canonical basis to canonical basis is $\widetilde{LM}$, then for $\forall \vec{x} \in \mathbb{R}^n$ under canonical basis we have

$$\vec{y} = \widetilde{LM}(\vec{x}) \tag{1}$$

, where $\vec{y}$ is the coordinate under canonical basis. Now we given the new basis $\vec{u}_i$, we have $\vec{y} = U\vec{y}^*$ and $\vec{x} = U\vec{x}^*$ where $\vec{x}^*$ and $\vec{y}^*$ are coordinates under basis U. Plug into (1) we could get $U\vec{y}^* = \widetilde{LM}(U\vec{x}^*)$. By the property of linear transformation, $U\vec{y}^* = \widetilde{LM}(U\vec{x}^*) = \widetilde{LM}U\vec{x}^*$. Multiply on both sides by $U^{\top}$ we get $\vec{y}^* = U^{\top}\widetilde{LM}U\vec{x}^*$. Meanwhile, by the property of composite linear transformation we have $\vec{y}^* = U^{\top}\widetilde{L}\widetilde{M}U\vec{x}^* (\widetilde{LM} = \widetilde{L}\widetilde{M})$.

Now we perform a trick $\vec{y}^* = U^{\top}\widetilde{L}\widetilde{M}U\vec{x}^* = U^{\top}\widetilde{L}UU^{\top}\widetilde{M}U\vec{x}^*$ Thus we could write $\widetilde{LM}' = \tilde{L}'\tilde{M}' = \tilde{M}'\tilde{L}'$.

Problem 5.4 (3 points)**.** *In this problem, we will see how to compress, by using a particular orthonormal basis called a "discrete cosine basis".*
All the questions are in the jupyter notebook `DCT.ipynb` *and have to be answered directly in the notebook. (Submit only a pdf export of your notebook: Print → Save as pdf)*
You have to use `Python` *and its library* `numpy`*. A useful command:* `A @ B` *: performs the matrix product of the matrix* `A` *with the matrix* `B`*.*

I attached two links below, one for pdf output and one for jupyter notebook link.
Jupyter notebook pdf link
Jupyter notebook link

Compressing images with Discrete Cosine Basis

In [1]:

```
%matplotlib inline
import numpy as np
import scipy.fftpack
import scipy.misc
import matplotlib.pyplot as plt
plt.gray()
```

```
<Figure size 640x480 with 0 Axes>
```

In [2]:

```
# Two auxiliary functions that we will use. You do not need to read them (but make sure to run tl

def dct(n):
    return scipy.fftpack.dct(np.eye(n), norm='ortho')

def plot_vector(v, color='k'):
    plt.plot(v,linestyle='', marker='o',color=color)
```

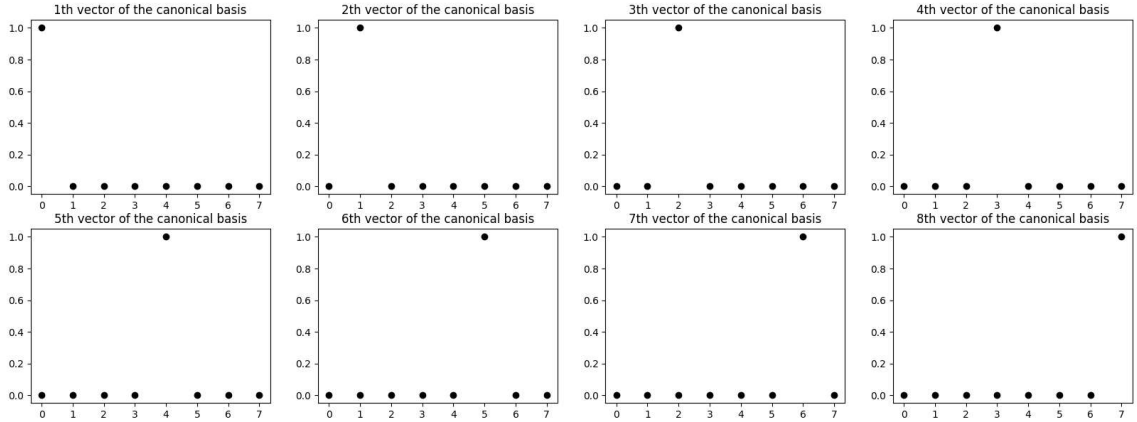
5.3.1 The canonical basis

The vectors of the canonical basis are the columns of the identity matrix in dimension n. We plot their coordinates below for $n = 8$.

In [3]:

```
identity = np.identity(8)
print(identity)

plt.figure(figsize=(20,7))
for i in range(8):
    plt.subplot(2,4,i+1)
    plt.title(f"{i+1}th vector of the canonical basis")
    plot_vector(identity[:,i])
```

```
[[1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1.]]
```



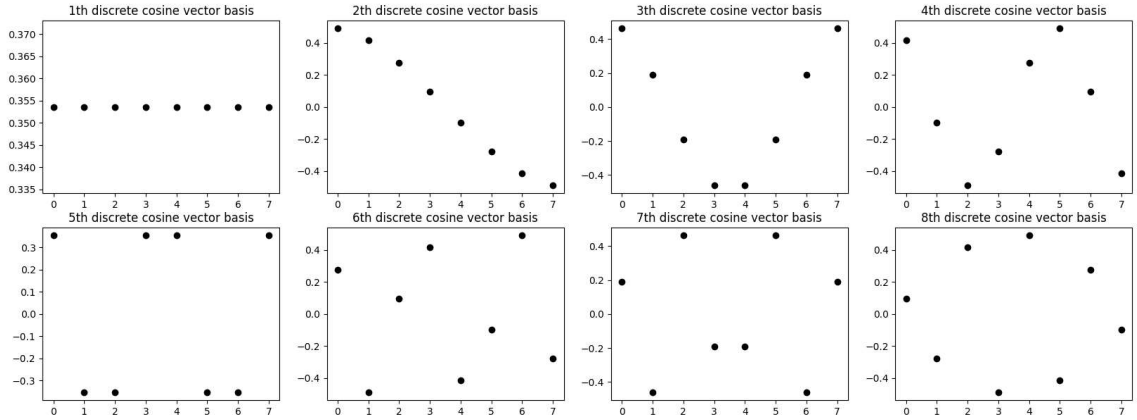
5.3.2 Discrete Cosine basis

The discrete cosine basis is another basis of $\mathbb{R}^n$. The function `dct(n)` outputs a square matrix of dimension n whose columns are the vectors of the discrete cosine basis.

In [4]:

```
# Discrete Cosine Transform matrix in dimension n = 8
D8 = dct(8)
print(np.round(D8,3))

plt.figure(figsize=(20,7))

for i in range(8):
    plt.subplot(2,4,i+1)
    plt.title(f"{i+1}th discrete cosine vector basis")
    plot_vector(D8[:,i])
```

```
[[ 0.354  0.49   0.462  0.416  0.354  0.278  0.191  0.098]
 [ 0.354  0.416  0.191 -0.098 -0.354 -0.49  -0.462 -0.278]
 [ 0.354  0.278 -0.191 -0.49  -0.354  0.098  0.462  0.416]
 [ 0.354  0.098 -0.462 -0.278  0.354  0.416 -0.191 -0.49 ]
 [ 0.354 -0.098 -0.462  0.278  0.354 -0.416 -0.191  0.49 ]
 [ 0.354 -0.278 -0.191  0.49  -0.354 -0.098  0.462 -0.416]
 [ 0.354 -0.416  0.191  0.098 -0.354  0.49  -0.462  0.278]
 [ 0.354 -0.49   0.462 -0.416  0.354 -0.278  0.191 -0.098]]
```



5.3 (a) Check numerically (in one line of code) that the columns of `D8` are an orthonormal basis of $\mathbb{R}^8$ (ie verify that the discrete cosine basis is an orthonormal basis).

In [13]:

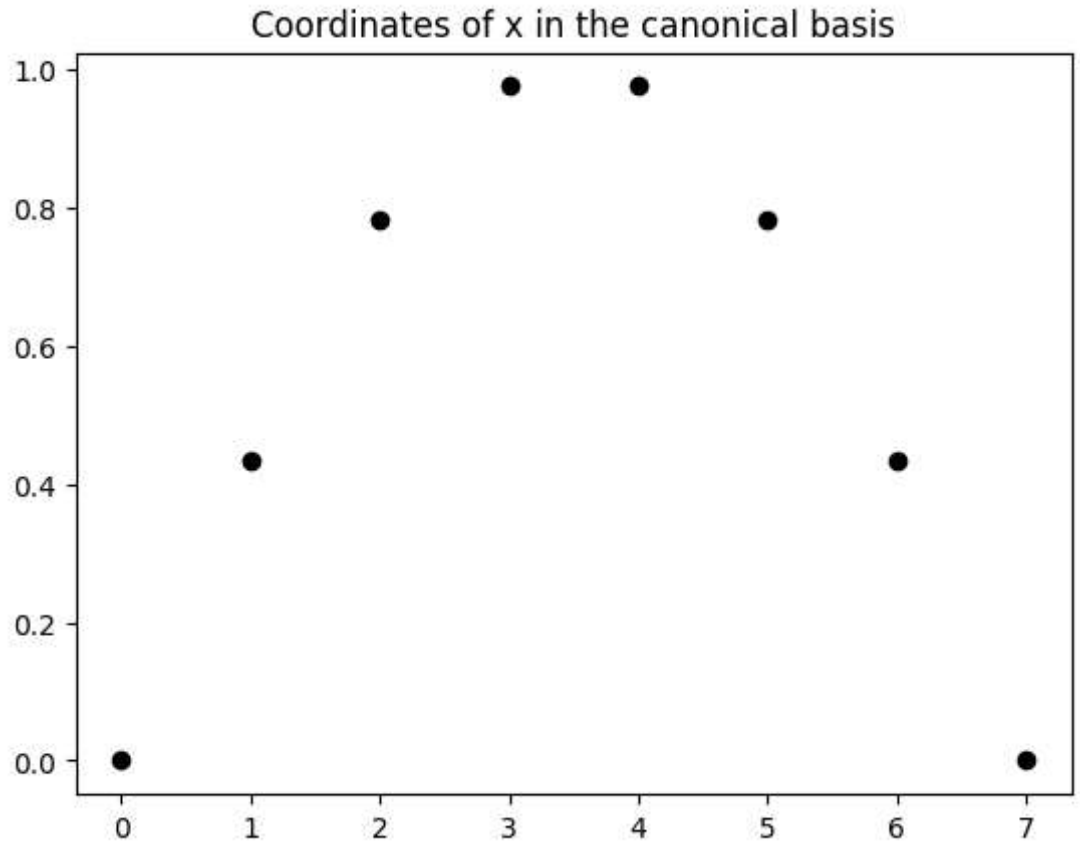
```
# Your answer here
D8.T @ D8 == np.eye(8)
```

Out[13]:

```
array([[False, False,  True,  True,  True,  True,  True,  True],
       [False, False, False, False,  True, False,  True,  True],
       [ True, False, False,  True,  True,  True,  True, False],
       [ True, False,  True, False,  True, False,  True, False],
       [ True,  True,  True,  True, False,  True, False, False],
       [ True, False,  True, False,  True, False, False, False],
       [ True,  True,  True,  True, False, False,  True, False],
       [ True,  True, False, False, False, False, False, False]])
```

In [14]:

```
# Let consider the following vector x
x = np.sin(np.linspace(0,np.pi,8))
plt.title('Coordinates of x in the canonical basis')
plot_vector(x)
```

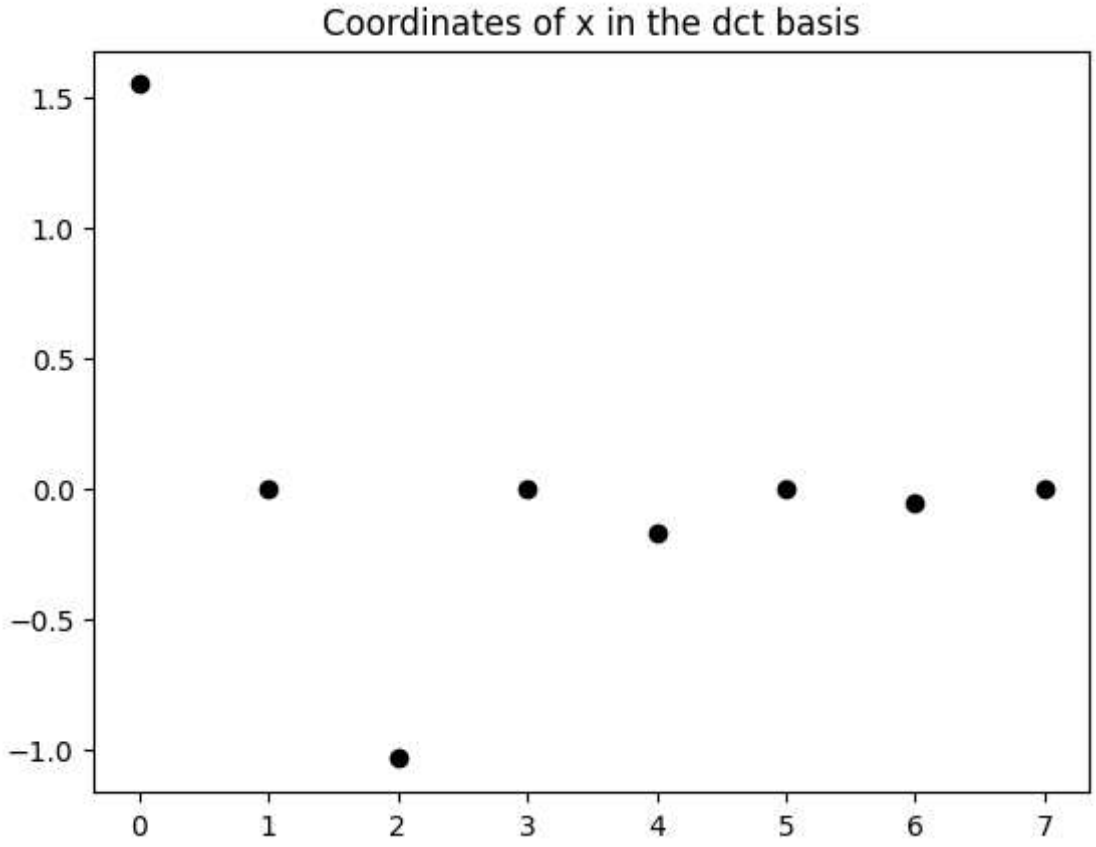


5.3 (b) Compute the vector $v \in \mathbb{R}^8$ of DCT coefficients of x. (1 line of code!), and plot them.

How can we obtain back x from v ? (1 line of code!).

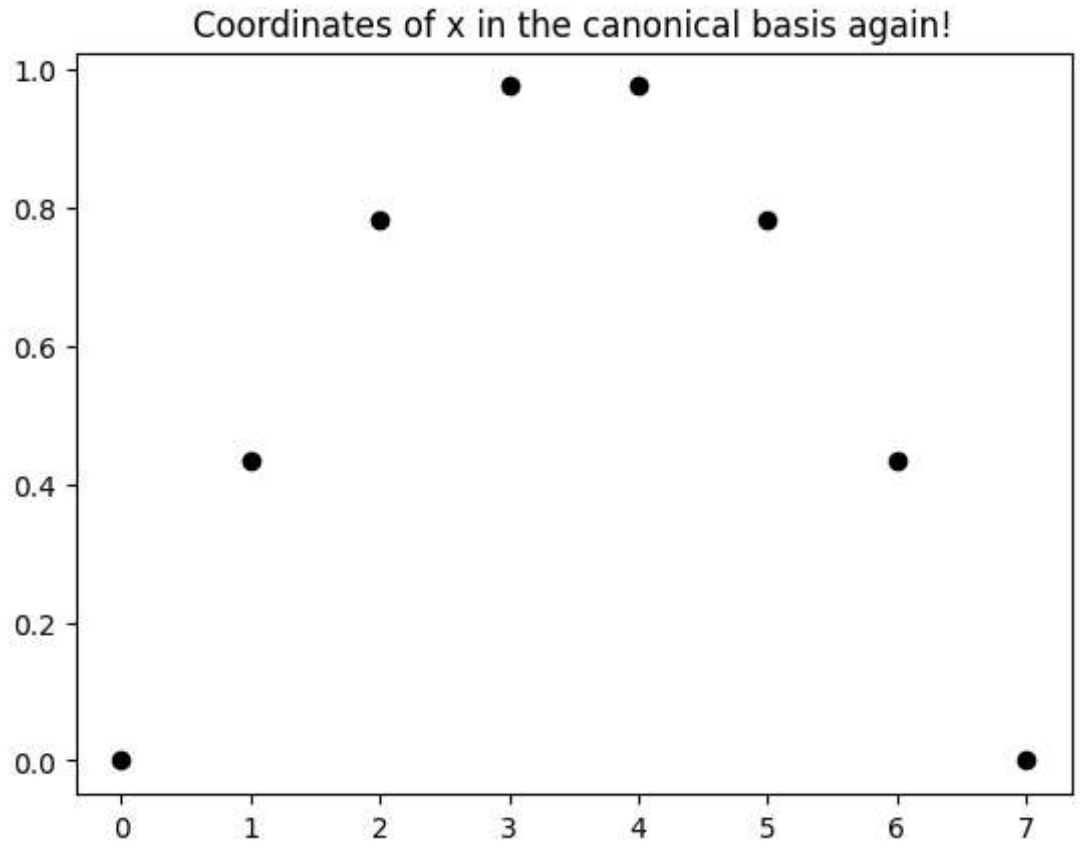
In [17]:

```
# Write your answer here
x_dct = D8.T @ x

plt.title('Coordinates of x in the dct basis')
plot_vector(x_dct)
```



In [18]:

```
x = D8 @ x_dct

plt.title('Coordinates of x in the canonical basis again!')
plot_vector(x)
```



5.3.3 Image compression

In this section, we will use DCT modes to compress images. Let's use one of the template images of python.

In [24]:

```
image = scipy.misc.face(gray=True)
h,w = image.shape
print(f'Height: {h}, Width: {w}')

plt.imshow(image)
```

```
Height: 768, Width: 1024

C:\Users\alexm\AppData\Local\Temp\ipykernel_20444\3842338847.py:1: DeprecationWar
ning: scipy.misc.face has been deprecated in SciPy v1.10.0; and will be completel
y removed in SciPy v1.12.0. Dataset methods have moved into the scipy.datasets mo
dule. Use scipy.datasets.face instead.
  image = scipy.misc.face(gray=True)
```

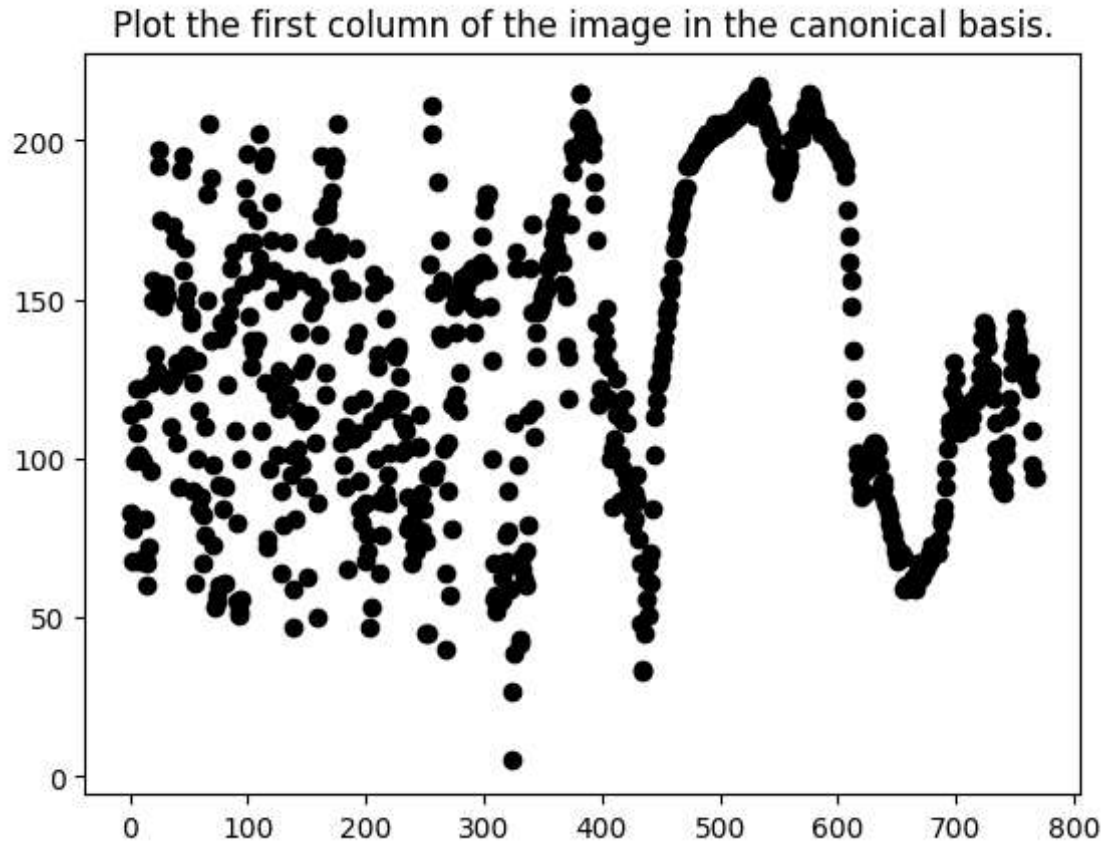
Out[24]:

```
<matplotlib.image.AxesImage at 0x1c298b90460>
```

5.3 (c) We will see each column of pixels as a vector in $\mathbb{R}^{768}$, and compute their coordinates in the DCT basis of $\mathbb{R}^{768}$. Plot the entries of x, the first column of our image.

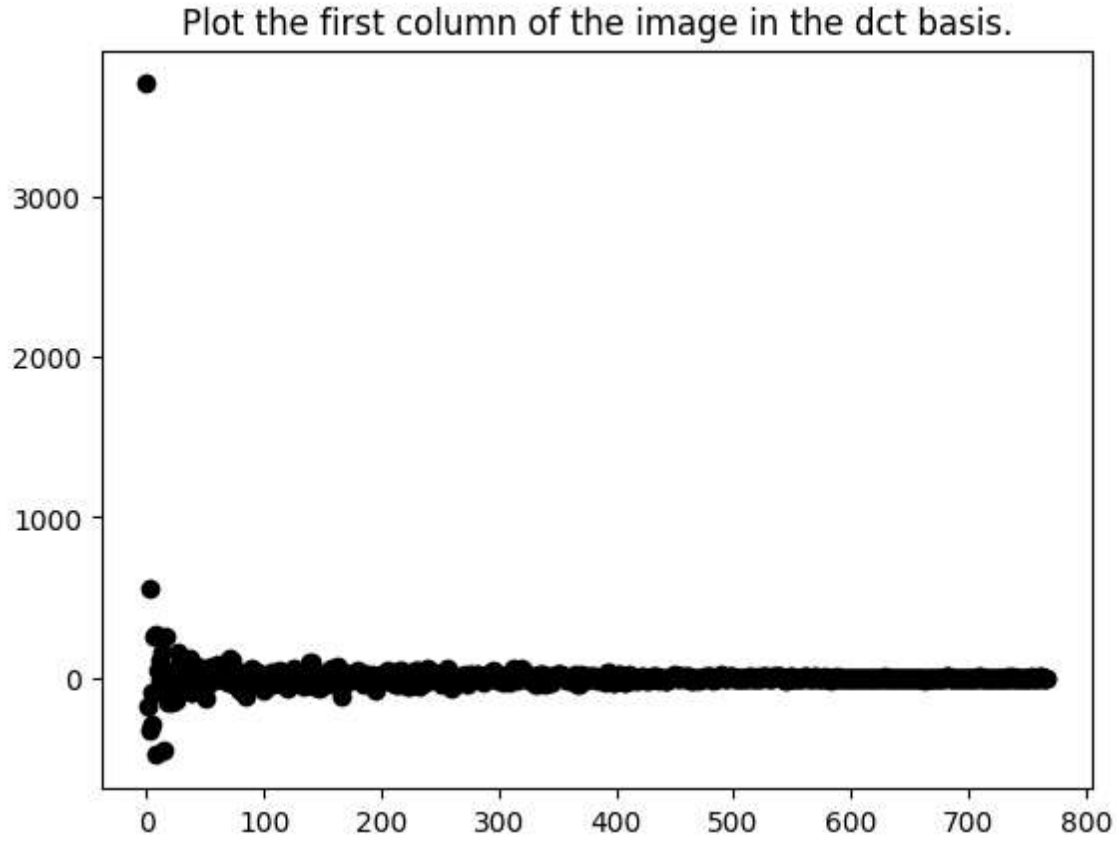
In [44]:

```
# Your answer here
plt.title('Plot the first column of the image in the canonical basis.')
plot_vector(image[:,0])
```



In [45]:

```
D768 = dct(768)
plt.title('Plot the first column of the image in the dct basis.')
plot_vector(D768.T @ image[:,0])
```



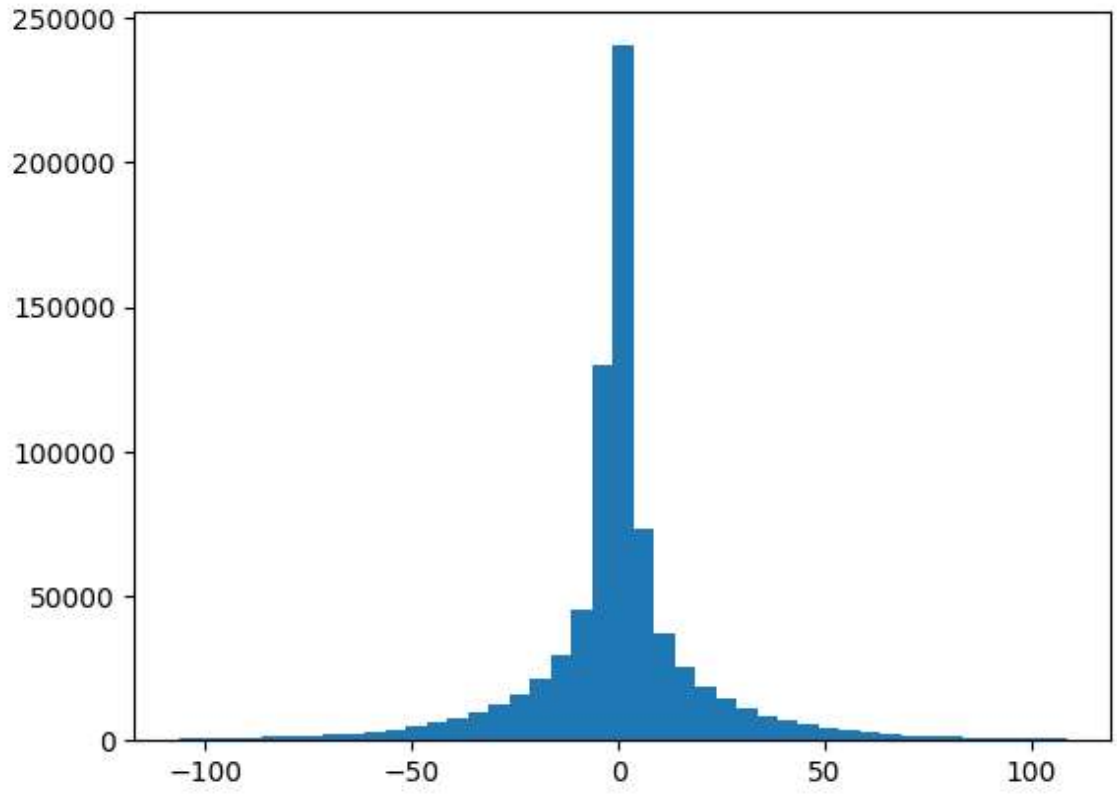
5.3 (d) Compute the 768 x 1024 matrix `dct_coeffs` whose columns are the DCT coefficients of the columns of `image` . Plot a histogram of their magnitudes using `plt.hist` .

In [65]:

```
# Your answer here
dct_coeffs = D768.T @ image
dct_coeffs_flat = dct_coeffs.flatten()
data_min = np.min(dct_coeffs_flat) / 10
data_max = np.max(dct_coeffs_flat) / 40
w = 5
bin_edges = np.arange(start=data_min, stop=data_max + w, step=w)
plt.hist(dct_coeffs.flatten(),bins=bin_edges)
```

Out[65]:

```
(array([   689.,    844.,    890.,    973.,   1243.,   1452.,   1717.,
          2170.,   2495.,   3000.,   3809.,   4762.,   6095.,   7556.,
          9500.,  12470.,  16127.,  21415.,  29855.,  45333., 130007.,
        240372.,  73255.,  36979.,  25785.,  18809.,  14493.,  11133.,
          8702.,   6732.,   5336.,   4294.,   3511.,   2785.,   2268.,
          1817.,   1582.,   1333.,   1111.,   1041.,    817.,    709.,
           648.]),
 array([-106.44312388, -101.44312388,  -96.44312388,  -91.44312388,
         -86.44312388,  -81.44312388,  -76.44312388,  -71.44312388,
         -66.44312388,  -61.44312388,  -56.44312388,  -51.44312388,
         -46.44312388,  -41.44312388,  -36.44312388,  -31.44312388,
         -26.44312388,  -21.44312388,  -16.44312388,  -11.44312388,
          -6.44312388,   -1.44312388,    3.55687612,    8.55687612,
          13.55687612,   18.55687612,   23.55687612,   28.55687612,
          33.55687612,   38.55687612,   43.55687612,   48.55687612,
          53.55687612,   58.55687612,   63.55687612,   68.55687612,
          73.55687612,   78.55687612,   83.55687612,   88.55687612,
          93.55687612,   98.55687612,  103.55687612,  108.55687612]),
 <BarContainer object of 43 artists>)
```



Since a large fraction of the DCT coefficients seems to be negligible, we see that the vector x can be well approximated by a linear combination of a small number of discrete cosines vectors.

Hence, we can "compress" the image by only storing the few DCT coefficients with the largest magnitude.

For instance, to reduce the size by 98%, we store only the top 2% largest (in absolute value) coefficients of `wavelet_coeffs`.

5.3 (e) Compute a matrix `thres_coeffs` which is the matrix `dct_coeffs` where about 98% smallest entries have been put to 0.

In [69]:

```
# Your answer here
# Flatten and sort the matrix(in the ascending order)
sorted_coeffs = np.sort(np.abs(dct_coeffs).flatten())

# Input the threshold
threshold_index = int(0.98 * len(sorted_coeffs))
threshold_value = sorted_coeffs[threshold_index]

# Set values below the threshold to 0
thres_coeffs = np.where(np.abs(dct_coeffs) < threshold_value, 0, dct_coeffs)

thres_coeffs
```

Out[69]:

```
array([[3697.60371463, 3687.50008492, 3683.24212669, ..., 2767.52851536,
        2777.70431385, 2782.39528479],
       [-170.00988   , -147.53068464,    0.        , ..., -294.3850494 ,
        -295.07337131, -294.44949706],
       [-320.49471312, -322.25987969, -322.86057333, ...,  155.39217275,
         159.72835212,  162.24937077],
       ...,
       [   0.        ,    0.        ,    0.        , ...,    0.        ,
           0.        ,    0.        ],
       [   0.        ,    0.        ,    0.        , ...,    0.        ,
           0.        ,    0.        ],
       [   0.        ,    0.        ,    0.        , ...,    0.        ,
           0.        ,    0.        ]])
```

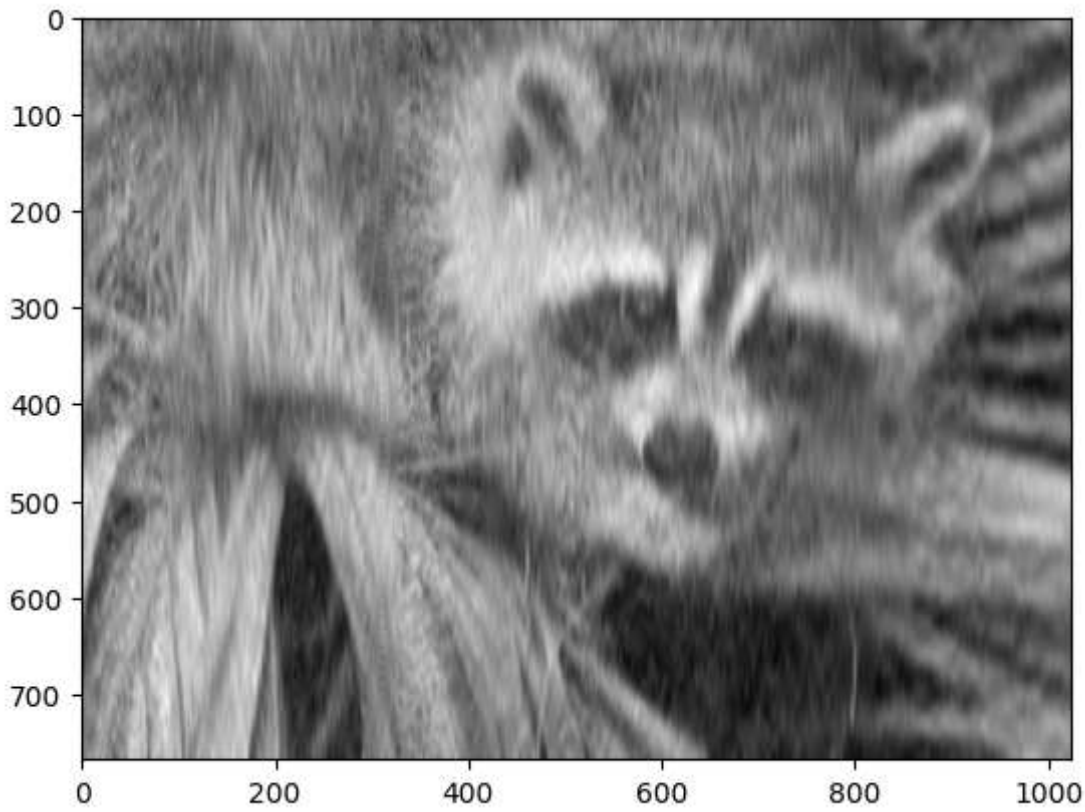
5.3 (f) Compute and plot the `compressed_image` corresponding to `thres_coeffs`.

In [74]:

```
# Your answer here
plt.imshow(D768 @ thres_coeffs)
```

Out[74]:

<matplotlib.image.AxesImage at 0x1c2a4c584c0>



Problem 5.5 ($\star$)**.** *Let S be a subspace of $\mathbb{R}^n$. We define the orthogonal complement of S by*

$$S^\perp \stackrel{\text{def}}{=} \{x \in \mathbb{R}^n \,|\, x \perp S\} = \{x \in \mathbb{R}^n \,|\, \forall y \in S,\, \langle x, y\rangle = 0\}.$$

One can prove that $S^\perp$ is a subspace of $\mathbb{R}^n$ (you do not have to prove it here).

(**a**) *Let P_S be the orthogonal projection on S. By applying the rank-nullity theorem to P_S, show that $\dim(S^\perp) = n - \dim(S)$.*

We want to show that $dim(S) = dim(Col(P_S))$ and $dim(S^\perp) = dim(\mathcal{N}(P_S))$.

- *(a) We first prove $S = Col(P_S)$. And we prove it using set equivalence. Note that $\forall \vec{x} \in S$, we have $P_S\vec{x} = \vec{x}$ by definition of the projection matrix P_S, which means $\vec{x} \in Col(P_S)$. For the other direction, we first denote an orthonormal basis for S, which is $B \in \mathbb{R}^{n\times k}$, then $P = BB^\top \in \mathbb{R}^{n\times n}$, and we have $Col(B) = S$ by definition of the basis. Then $\forall \vec{x} \in Col(P_S)$, we have $\vec{x} = BB^\top\vec{y} = B\vec{z} \in Col(B) = S$. Thus we have shown that $S = Col(P_S)$ and thus $dim(S) = dim(Col(P_S)) = rank(P_S)$.*
- *(b) We then prove $S^\perp = \mathcal{N}(P_S)$, this is obvious since $Col(P_S)^\perp = \mathcal{N}(P_S)$, which is true for any matrices.*

. Then we can apply the rank nullity theorem to P_S and get $dim(Col(P_S)) + dim(\mathcal{P}_S) = n$(since $P_S \in \mathbb{R}^{n\times n}$), thus $dim(S) + dim(S^\perp) = n$ as desired.

(**b**) *Show that $(S^\perp)^\perp = S$.*

We prove by set equivalence:

- *(a) We first show that $(S^\perp)^\perp \subseteq S$. $\forall \vec{x} \in (S^\perp)^\perp$, $\langle \vec{x}, \vec{y}\rangle = 0, \forall \vec{y} \in S^\perp$. Since $\forall \vec{x} \in S, \langle \vec{x}, \vec{y}\rangle = 0, \forall \vec{y} \in S^\perp$, we have that if $\vec{x} \in (S^\perp)^\perp$ then $\vec{x} \in S$.*
- *(b) We then show $S \subseteq (S^\perp)^\perp$. $\forall \vec{x} \in S$, we have $\langle \vec{x}, \vec{y}\rangle = 0, \forall \vec{y} \in S^\perp$, since $\vec{x}$ is orthogonal to every vector in $S^\perp$, we have $\vec{x} \in (S^\perp)^\perp$.*

, thus $(S^\perp)^\perp = S$ as desired.

(**c**) *Show that for any $x \in \mathbb{R}^n$, there exists a unique pair $(u, v) \in S \times S^\perp$ such that $x = u + v$.*

Suppose there is an orthonormal basis for S which is $\{\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_k\}$ where $\|\vec{v}_i\|_2 = 1$.

- *(a) We first show the existence: $\forall \vec{x} \in \mathbb{R}^n$, we have $\vec{x} = \langle \vec{x}, \vec{v}_1\rangle\vec{v}_1 + \langle \vec{x}, \vec{v}_2\rangle\vec{v}_2 + \cdots + \langle \vec{x}, \vec{v}_k\rangle\vec{v}_k + \vec{x} - \langle \vec{x}, \vec{v}_1\rangle\vec{v}_1 + \langle \vec{x}, \vec{v}_2\rangle\vec{v}_2 + \cdots + \langle \vec{x}, \vec{v}_k\rangle\vec{v}_k$. We denote $\vec{u} = \langle \vec{x}, \vec{v}_1\rangle\vec{v}_1 + \langle \vec{x}, \vec{v}_2\rangle\vec{v}_2 + \cdots + \langle \vec{x}, \vec{v}_k\rangle\vec{v}_k$ and $\vec{v} = \vec{x} - \langle \vec{x}, \vec{v}_1\rangle\vec{v}_1 + \langle \vec{x}, \vec{v}_2\rangle\vec{v}_2 + \cdots + \langle \vec{x}, \vec{v}_k\rangle\vec{v}_k$. We observe that $\vec{u} \in S$ for sure by definition of the orthonormal basis. Since $\langle \vec{v}, \vec{v}_i\rangle = \langle \vec{x}, \vec{v}_i\rangle - \langle \vec{x}, \vec{v}_i\rangle\langle \vec{v}_i, \vec{v}_i\rangle = 0$, we know that $\vec{v} \perp \vec{v}_i, \forall i = 1, 2, \cdots, k$. Thus $\vec{v} \perp Span(\{\vec{v}_1, \cdots, \vec{v}_k\}) = S$, which implies $\vec{v} \in S^\perp$. Thus we have proved that $\vec{x} = \vec{u} + \vec{v}$ with desired properties exists.*
- *(b) We then show the uniqueness: Suppose there exists two representation such that $\vec{x} = \vec{u}_1 + \vec{v}_1 = \vec{u}_2 + \vec{v}_2$ thus $\vec{0} = (\vec{u}_1 - \vec{u}_2) + (\vec{v}_1 - \vec{v}_2)$ where $\vec{u}_1 \neq \vec{u}_2$ and $\vec{v}_1 \neq \vec{v}_2$. Since S and $S^\perp$ are both subspaces, we have $\vec{u}_1 - \vec{u}_2 \in S$ and $\vec{v}_1 - \vec{v}_2 \in S^\perp$ and we know that they must be orthogonal to each other.But $\vec{0} = (\vec{u}_1 - \vec{u}_2) + (\vec{v}_1 - \vec{v}_2)$implies that $(\vec{u}_1 - \vec{u}_2) = -(\vec{v}_1 - \vec{v}_2)$ and that $-(\vec{v}_1 - \vec{v}_2)^\top(\vec{v}_1 - \vec{v}_2) = 0$ and thus $\|\vec{v}_1 - \vec{v}_2\|_2 = 0$, which implies that $\vec{v}_1 = \vec{v}_2$ by the property of l2 norm. Similarly we could get $\vec{u}_1 = \vec{u}_2$. Thus this representation must be unique.*