DS-GA 1014 - Homework 4

Eric Niblock

September 26th, 2020

1. (2 points). Let $||\cdot||$ be the usual Euclidean norm on $\mathbb{R}^n$. For $\vec{\mathbf{x}} \in \mathbb{R}^n$ compute (and justify your result):

$$max\big\{\vec{\mathbf{v}}^T\vec{\mathbf{x}} \mid \vec{\mathbf{v}} \in \mathbb{R}^n, ||\vec{\mathbf{v}}|| = 1\big\} \tag{1}$$

First, we employ the following theorem given below,

The Cauchy–Schwarz inequality states that for all vectors $\vec{\boldsymbol{u}}$ and $\vec{\boldsymbol{v}}$ of an inner product space it is true that

$$\langle\vec{\mathbf{u}}, \vec{\mathbf{v}}\rangle \leq ||\vec{\mathbf{u}}||\,||\vec{\mathbf{v}}|| \tag{2}$$

Moreover, there is equality if and only if $\vec{\boldsymbol{u}}$ and $\vec{\boldsymbol{v}}$ are linearly dependent. [Thm. 1]

Given that $\vec{\mathbf{v}}^T\vec{\mathbf{x}} = \langle\vec{\mathbf{v}}, \vec{\mathbf{x}}\rangle$, then we have that,

$$\langle\vec{\mathbf{v}}, \vec{\mathbf{x}}\rangle \leq ||\vec{\mathbf{v}}||\,||\vec{\mathbf{x}}|| = ||\vec{\mathbf{x}}|| \tag{3}$$

Since we have that $||\vec{\mathbf{v}}|| = 1$. So, the set of all max values is such that,

$$\langle\vec{\mathbf{v}}, \vec{\mathbf{x}}\rangle = \vec{\mathbf{v}}^T\vec{\mathbf{x}} = ||\vec{\mathbf{x}}|| \tag{4}$$

Furthermore, by *Theorem 1*, we know that strict equality comes about only when $\overrightarrow{\mathbf{v}} = \lambda \overrightarrow{\mathbf{x}}$, so,

$$\overrightarrow{\mathbf{v}}^T \overrightarrow{\mathbf{x}} = \lambda \overrightarrow{\mathbf{x}}^T \overrightarrow{\mathbf{x}} = \lambda ||\overrightarrow{\mathbf{x}}||^2 = ||\overrightarrow{\mathbf{x}}|| \tag{5}$$

$$\lambda = \frac{1}{||\overrightarrow{\mathbf{x}}||} \tag{6}$$

So the max values that are generated are $||\overrightarrow{\mathbf{x}}||$, and they occur for any $\overrightarrow{\mathbf{x}} \in \mathbb{R}^{\mathrm{n}}$ when we take $\overrightarrow{\mathbf{v}}$ to be,

$$\overrightarrow{\mathbf{v}} = \frac{\overrightarrow{\mathbf{x}}}{||\overrightarrow{\mathbf{x}}||} \tag{7}$$

2. (1 points). Show that for all $x \in \mathbb{R}^n$,

$$\frac{1}{\sqrt{n}}||\overrightarrow{\mathbf{x}}||_1 \leq ||\overrightarrow{\mathbf{x}}||_2 \leq ||\overrightarrow{\mathbf{x}}||_1 \tag{8}$$

First we show that $||\overrightarrow{\mathbf{x}}||_2 \leq ||\overrightarrow{\mathbf{x}}||_1$. We know that,

$$||\overrightarrow{\mathbf{x}}||_2^2 = \sum_{i=1}^{n} x_i^2 \leq \left(\sum_{i=1}^{n} x_i^2 + 2\sum_{i \neq j}^{n} x_i x_j \right) = ||\overrightarrow{\mathbf{x}}||_1^2 \tag{9}$$

$$||\overrightarrow{\mathbf{x}}||_2^2 \leq ||\overrightarrow{\mathbf{x}}||_1^2 \implies ||\overrightarrow{\mathbf{x}}||_2 \leq ||\overrightarrow{\mathbf{x}}||_1 \tag{10}$$

Since every term above is guaranteed to be positive. So we have shown the latter half of the inequality, $||\overrightarrow{\mathbf{x}}||_2 \leq ||\overrightarrow{\mathbf{x}}||_1$. Now, we know that,

$$\frac{1}{\sqrt{n}}||\overrightarrow{\mathbf{x}}||_1 \leq ||\overrightarrow{\mathbf{x}}||_2 \qquad \iff \qquad ||\overrightarrow{\mathbf{x}}||_1 \leq \sqrt{n}||\overrightarrow{\mathbf{x}}||_2 \tag{11}$$

We have that the following holds true,

$$||\overrightarrow{\mathbf{x}}||_1 = \sum_{i=1}^{n} |x_i| = \sum_{i=1}^{n} |x_i| \cdot 1 \tag{12}$$

At this point, we define the vector $\overrightarrow{\mathbf{x}}$ such that $\overrightarrow{\mathbf{x}} = (x_1, ..., x_n)$ and $\overrightarrow{\mathbf{y}}$ such that $\overrightarrow{\mathbf{y}} = (1, ..., 1)$ for length n. Then by *Theorem 1*,

$$\sum_{i=1}^{n} |x_i| \leq \sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} 1} = \sqrt{n}||\overrightarrow{\mathbf{x}}||_2 \tag{13}$$

And finally, we produce the desired result,

$$||\overrightarrow{\mathbf{x}}||_1 \leq \sqrt{n}||\overrightarrow{\mathbf{x}}||_2 \qquad \Longrightarrow \qquad \frac{1}{\sqrt{n}}||\overrightarrow{\mathbf{x}}||_1 \leq ||\overrightarrow{\mathbf{x}}||_2 \tag{14}$$

So, therefore we have shown that,

$$\frac{1}{\sqrt{n}}||\overrightarrow{\mathbf{x}}||_1 \leq ||\overrightarrow{\mathbf{x}}||_2 \leq ||\overrightarrow{\mathbf{x}}||_1 \tag{15}$$

3. (3 points). Let S be a subspace of $\mathbb{R}^n$. We define the orthogonal complement of S by

$$S^{\perp} = \left\{\overrightarrow{\mathbf{x}} \in \mathbb{R}^n \,|\, \overrightarrow{\mathbf{x}} \perp S\right\} = \left\{\overrightarrow{\mathbf{x}} \in \mathbb{R}^n \,|\, \forall \overrightarrow{\mathbf{y}} \in S, \langle \overrightarrow{\mathbf{x}}, \overrightarrow{\mathbf{y}} \rangle = 0\right\} \tag{16}$$

(a) Show that $S^{\perp}$ is a subspace of $\mathbb{R}^n$.

In order to show that $S^{\perp}$ is a subspace, we must show that $S^{\perp}$ (1) contains the zero vector, (2) is closed under vector-addition, and (3) is closed under scalar multiplication.

We know that $\overrightarrow{\mathbf{0}} \in S^{\perp}$, since for all $\overrightarrow{\mathbf{y}} \in S$, we have $\langle \overrightarrow{\mathbf{0}}, \overrightarrow{\mathbf{y}} \rangle = 0$. So, condition (1) is satisfied.

If we have two vectors such that $\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{v}} \in S^{\perp}$, then we must show that $\overrightarrow{\mathbf{u}} + \overrightarrow{\mathbf{v}} \in S^{\perp}$. Since $\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{v}} \in S^{\perp}$, we know that $\langle \overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{y}} \rangle = 0$ and that $\langle \overrightarrow{\mathbf{v}}, \overrightarrow{\mathbf{y}} \rangle = 0$, for all $\overrightarrow{\mathbf{y}} \in S$. If we are to have $\overrightarrow{\mathbf{u}} + \overrightarrow{\mathbf{v}} \in S^{\perp}$ then we must show $\langle \overrightarrow{\mathbf{u}} + \overrightarrow{\mathbf{v}}, \overrightarrow{\mathbf{y}} \rangle = 0$, for all $\overrightarrow{\mathbf{y}} \in S$. But, by the properties of the inner product, we know that,

$$\langle \overrightarrow{\mathbf{u}} + \overrightarrow{\mathbf{v}}, \overrightarrow{\mathbf{y}} \rangle = \langle \overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{y}} \rangle + \langle \overrightarrow{\mathbf{v}}, \overrightarrow{\mathbf{y}} \rangle = 0 + 0 = 0 \tag{17}$$

So $\overrightarrow{\mathbf{u}} + \overrightarrow{\mathbf{v}} \in S^{\perp}$. This satisfies condition (2).

If we have a vector, $\overrightarrow{\mathbf{u}} \in S^{\perp}$, then we must show that $c\overrightarrow{\mathbf{u}} \in S^{\perp}$ for any $c \in \mathbb{R}$. Since $\overrightarrow{\mathbf{u}} \in S^{\perp}$, we know that $\langle \overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{y}} \rangle = 0$ for all $\overrightarrow{\mathbf{y}} \in S$. We must show that $\langle c\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{y}} \rangle = 0$ for any $c \in \mathbb{R}$, and for all $\overrightarrow{\mathbf{y}} \in S$. But, by the properties of the inner product, we have that,

$$\langle c\overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{y}} \rangle = c\langle \overrightarrow{\mathbf{u}}, \overrightarrow{\mathbf{y}} \rangle = c(0) = 0 \tag{18}$$

This shows that $c\overrightarrow{\mathbf{u}} \in S^{\perp}$ for any $c \in \mathbb{R}$. Therefore condition (3) is satisfied.

Having satisfied all of the conditions for a subspace, we note that $S^{\perp}$ is a subspace.

(b) Show that $dim(S^{\perp}) = n - dim(S)$. Hint: use the rank-nullity theorem.

Take the family of vectors $(\overrightarrow{\mathbf{v}_1}, ..., \overrightarrow{\mathbf{v}_m})$ to be a set of basis vectors for the subspace S, with all $\overrightarrow{\mathbf{v}_i} \in \mathbb{R}^n$. Furthermore, take the family of vectors $(\overrightarrow{\mathbf{v}_{m+1}}, ..., \overrightarrow{\mathbf{v}_k})$ to be a set of basis vectors for the subspace $S^\perp$, with all $\overrightarrow{\mathbf{v}_i} \in \mathbb{R}^n$. If this is the case, then we have that $dim(S) = m$ and $dim(S^\perp) = k - m$. So if we hope to show $dim(S^\perp) = n - dim(S)$, then we must show $k - m = n - m \implies k = n$. We now attempt to show that $k = n$.

First, we know that the family of vectors $(\overrightarrow{\mathbf{v}_1}, ..., \overrightarrow{\mathbf{v}_m}, \overrightarrow{\mathbf{v}_{m+1}}, ..., \overrightarrow{\mathbf{v}_k})$ is linearly independent, since for any basis vector $\overrightarrow{\mathbf{v}_i} \in S^\perp$ we have that $\overrightarrow{\mathbf{v}_i} \perp S$, and therefore $\overrightarrow{\mathbf{v}_i} \perp \overrightarrow{\mathbf{y}}$ for any vector $\overrightarrow{\mathbf{y}} \in S$ (furthermore, $\overrightarrow{\mathbf{v}_i}$ must be independent of all other basis vectors within $S^\perp$, by the definition of a basis). Since the family of vectors is linearly independent, this implies that $k \leq n$, since a vector space $V = \mathbb{R}^n$ can contain at most n linearly independent vectors.

Suppose that $k < n$. Then, if we form a matrix A, with its rows being formed of $\overrightarrow{\mathbf{v}_1^T}, ..., \overrightarrow{\mathbf{v}_k^T}$, the shape of A would be $k \times n$. Since $n > k$, the matrix is wide and has less than n pivots. This implies that the kernel of A contains at least one nonzero vector, call it $\overrightarrow{\mathbf{z}}$. Then,

$$A\overrightarrow{\mathbf{z}} = \begin{bmatrix} \overrightarrow{\mathbf{v}_1^T} \\ \vdots \\ \overrightarrow{\mathbf{v}_k^T} \end{bmatrix} \overrightarrow{\mathbf{z}} = \begin{bmatrix} \langle \overrightarrow{\mathbf{v}_1}, \overrightarrow{\mathbf{z}} \rangle \\ \vdots \\ \langle \overrightarrow{\mathbf{v}_k}, \overrightarrow{\mathbf{z}} \rangle \end{bmatrix} = \overrightarrow{\mathbf{0}} \tag{19}$$

Since $\langle \overrightarrow{\mathbf{v}_1}, \overrightarrow{\mathbf{z}} \rangle = ... = \langle \overrightarrow{\mathbf{v}_k}, \overrightarrow{\mathbf{z}} \rangle = 0$, we deduce that $\overrightarrow{\mathbf{z}} \in S$ and $\overrightarrow{\mathbf{z}} \in S^\perp$. However, then $\langle \overrightarrow{\mathbf{z}}, \overrightarrow{\mathbf{z}} \rangle = 0$, implying that $\overrightarrow{\mathbf{z}} = \overrightarrow{\mathbf{0}}$, which is a contradiction. Therefore, $k = n$. Since $k = n$ we have shown that $dim(S^\perp) = n - dim(S)$.

(c) Let $v = (1, 1, 1) \in \mathbb{R}^3$ and define

$$H = \left\{ \overrightarrow{\mathbf{x}} \in \mathbb{R}^3 \mid \overrightarrow{\mathbf{x}} \perp \overrightarrow{\mathbf{v}} \right\} = Span(v)^\perp \tag{20}$$

Find an orthonormal basis of H and an orthonormal basis of $H^\perp$.

We might instead rephrase the definition of H such that

$$H = \left\{ \overrightarrow{\mathbf{x}} \in \mathbb{R}^3 \mid \overrightarrow{\mathbf{v}}^T \overrightarrow{\mathbf{x}} = 0 \right\} \tag{21}$$

Thus the problem becomes finding the null space of $\vec{\mathbf{v}}^T$. Considering this vector as a matrix, it has already been row reduced, and we are left with,

$$x_1 + x_2 + x_3 = 0 \tag{22}$$

With x_2 and x_3 being free-variables. Thus, we suggest the basis,

$$\left\{ \begin{bmatrix} 2 \\ -1 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix} \right\} \tag{23}$$

Note that the vectors within the basis are both orthogonal with $\vec{\mathbf{v}}$ as well as with each other. Now, all we need to do is normalize them by dividing by their length. Thus, an orthonormal basis for H is given by,

$$\left\{ \frac{1}{\sqrt{6}} \begin{bmatrix} 2 \\ -1 \\ -1 \end{bmatrix}, \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix} \right\} \tag{24}$$

(d) Write the matrix of P_H, the orthogonal projection on H.

The matrix P_H is given by $P_H = AA^T$ where A has columns that represent the orthonormal basis vectors of H. So,

$$P_H = AA^T = \begin{bmatrix} 2/\sqrt{6} & 0 \\ -1/\sqrt{6} & 1/\sqrt{2} \\ -1/\sqrt{6} & -1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 2/\sqrt{6} & -1/\sqrt{6} & -1/\sqrt{6} \\ 0 & 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix} \tag{25}$$

$$P_H = \begin{bmatrix} 2/3 & -1/3 & -1/3 \\ -1/3 & 2/3 & -1/3 \\ -1/3 & -1/3 & 2/3 \end{bmatrix} \tag{26}$$

4. **(4 points). In this problem, we will see how to compress (using a method similar to the one used in the jpeg standard) and denoise images, by using a particular orthonormal basis called a "wavelet basis".**

   Please observe the attached PDF containing the Python file used for this problem.

In [44]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plot
plot.gray()
```

```
<Figure size 432x288 with 0 Axes>
```

In [45]:
```
# Two auxiliary functions that we will use. You do not need to read them (but
 make sure to run this cell!)

# Function that construct a matrix whose columns are the vectors of the Haar w
avelet basis in dimension n=2**d
def haar(d):
    n = 2**d
    m = np.zeros((n,n))
    l = 1
    for i in range(d):
        k = int(n / (2*l))
        for j in range(k):
            m[2*j*l : 2*j*l + l, j+n-2*k] = 1 / np.sqrt(2*l)
            m[2*j*l+l : 2*(j+1)*l, j+n-2*k, ] = -1 / np.sqrt(2*l)
        l = 2*l
    m[:,-1] = 1 / np.sqrt(n)
    return m

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

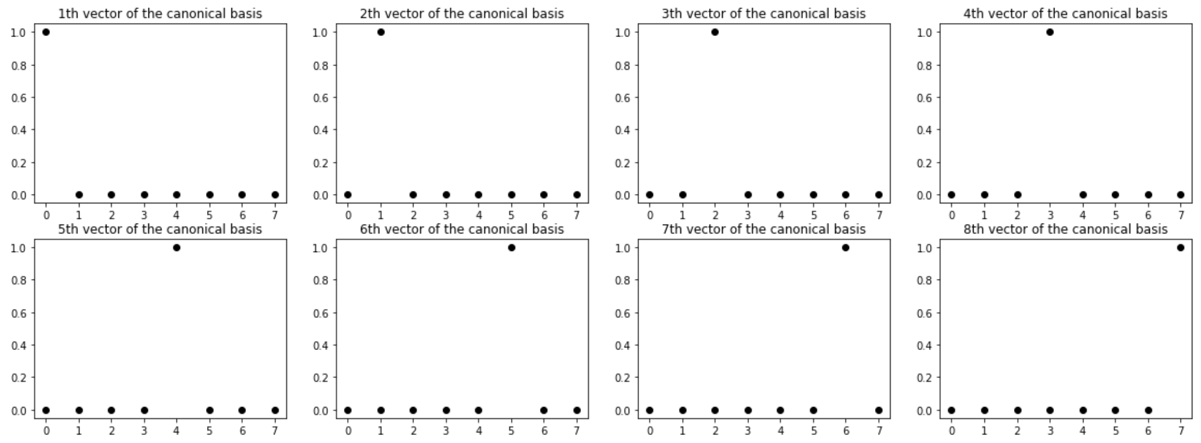
1. Haar Wavelets

1.1. 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 [46]:
```
identity = np.identity(8)
print(identity)

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

print('\n Nothing new so far...')
```

```
[[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.]]

 Nothing new so far...
```

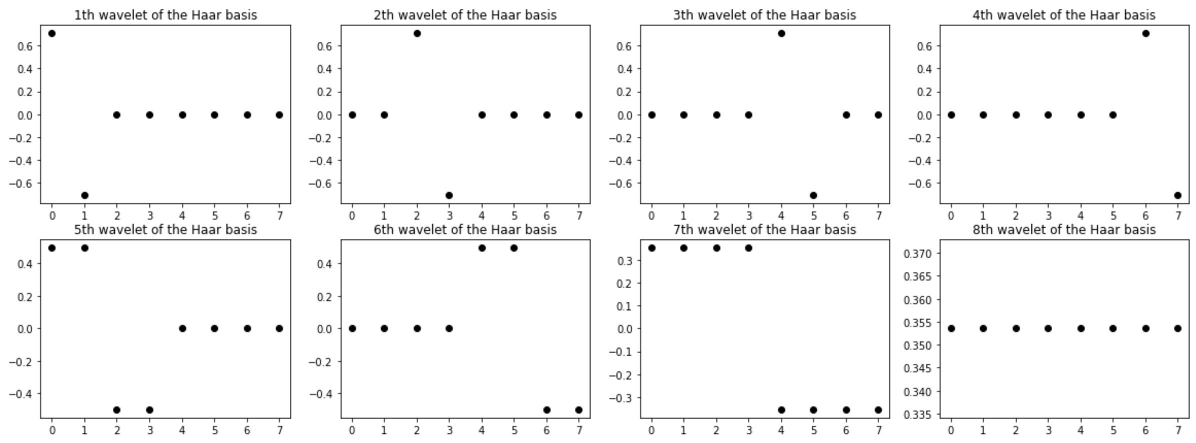


1.2. Haar wavelet basis

Haar wavelets basis is another basis of $\mathbb{R}^n$ when n is a power of 2, that is $n = 2^k$ for some k. The function `haar(k)` outputs a square matrix of dimension $n = 2^k$ whose columns are the vectors of the Haar wavelet basis.

In [47]:
```
# Matrix of Haar wavelets in dimension n = 2**3 = 8
H8 = haar(3)
print(np.round(H8,3))

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

for i in range(8):
    plot.subplot(2,4,i+1)
    plot.title(f"{i+1}th wavelet of the Haar basis")
    plot_vector(H8[:,i])
```

```
[[ 0.707  0.     0.     0.     0.5    0.     0.354  0.354]
 [-0.707  0.     0.     0.     0.5    0.     0.354  0.354]
 [ 0.     0.707  0.     0.    -0.5    0.     0.354  0.354]
 [ 0.    -0.707  0.     0.    -0.5    0.     0.354  0.354]
 [ 0.     0.     0.707  0.     0.     0.5   -0.354  0.354]
 [ 0.     0.    -0.707  0.     0.     0.5   -0.354  0.354]
 [ 0.     0.     0.     0.707  0.    -0.5   -0.354  0.354]
 [ 0.     0.     0.    -0.707  0.    -0.5   -0.354  0.354]]
```



(a) Check numerically (in one line of code) that the columns of `H8` are an orthonormal basis of $\mathbb{R}^8$ (ie verify that the Haar wavelet basis is an orthonormal basis).

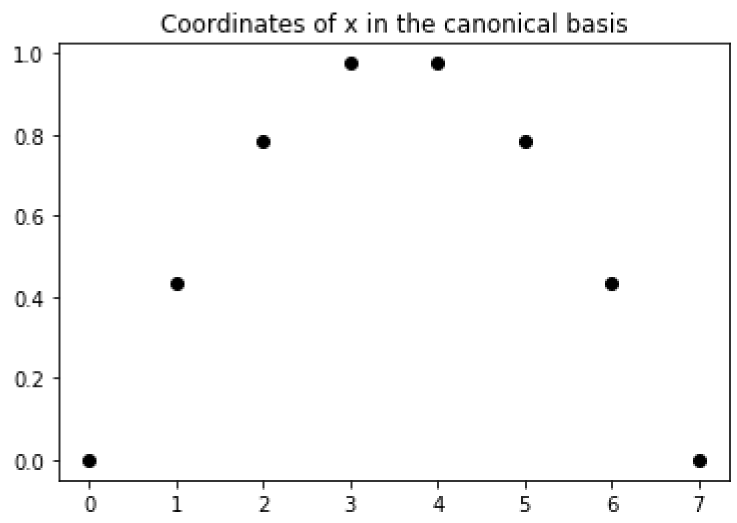
In [48]:
```
print(np.round(np.dot(H8, H8.T),10))
```

```
[[ 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.]]
```

The vectors of the Haar wavelet basis are called **wavelets**.

Given a vector $x \in \mathbb{R}^n$, the coordinates of x in the wavelet Haar basis are called the **wavelet coefficients** of x.

In [49]:
```
# Let consider the following vector x
x = np.sin(np.linspace(0,np.pi,8))
plot.title('Coordinates of x in the canonical basis')
plot_vector(x)
```



(b) Compute the vector $v \in \mathbb{R}^8$ of wavelet coefficients of x. (1 line of code!)

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

In [50]:
```
v = np.dot(H8.T, x)
print('v: ', v)

same_x = np.dot(H8, v)
print('x: ', same_x)
```

```
v:  [-3.06802134e-01 -1.36539795e-01  1.36539795e-01  3.06802134e-01
 -6.61437828e-01  6.61437828e-01 -6.99260413e-17  1.54901862e+00]
x:  [0.00000000e+00 4.33883739e-01 7.81831482e-01 9.74927912e-01
 9.74927912e-01 7.81831482e-01 4.33883739e-01 1.66533454e-16]
```

2. Image compression

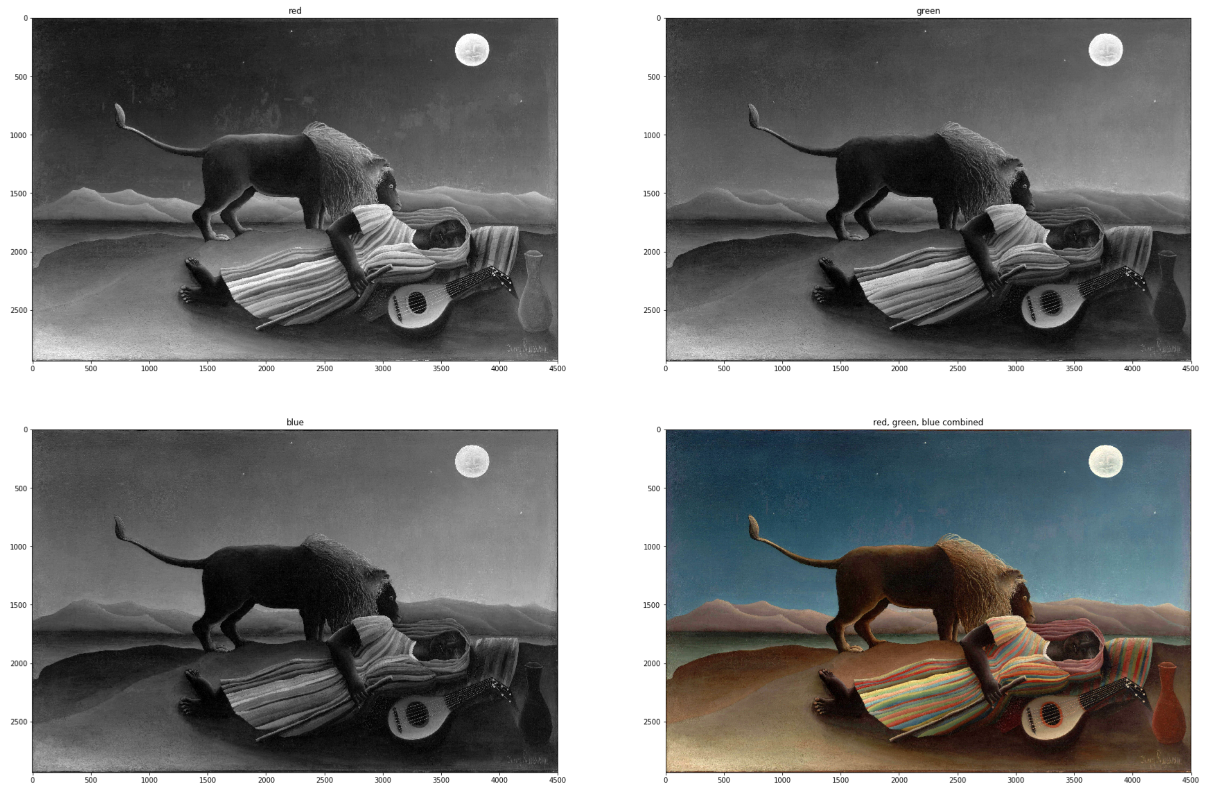
In this section, we will use Haar wavelets to compress images

In [51]:
```
# Reading the image file
img_RGB = plot.imread('sleeping.jpg')
h,w,d = img_RGB.shape
print(f'Height: {h}, Width: {w}, Number of channels: {d} (Red, Green, Blue)')
```

```
Height: 2934, Width: 4500, Number of channels: 3 (Red, Green, Blue)
```

The 'image tensor' `img_RGB` contains 3 matrices: `img_RGB[:,:,0]` , `img_RGB[:,:,1]` and `img_RGB[:,:,2]` , giving for each pixel the amount of red, green and blue.

```
In [52]: plot.figure(figsize=(30,20))
         colors = ['red','green','blue']
         for i in range(3):
             plot.subplot(2, 2, i+1)
             plot.title(colors[i])
             plot.imshow(img_RGB[:,:,i])
         plot.subplot(2,2,4)
         plot.title('red, green, blue combined')
         plot.imshow(img_RGB)
         plot.show()
```



Our goal is to compress and denoise such RGB images. **For simplicity, we will do it for only one of the three colors:**

```
In [53]: image = img_RGB[:,:,1] # Consider the green component only
```

It will be much more convenient to use images whose dimensions are powers of 2. Hence we crop the original image:

In [54]:
```
def power2crop(img):
    height, width= img.shape
    height2, width2 = 2**int(np.log2(height)), 2**int(np.log2(width))
    padh, padv= int((height-height2)/2), int((width-width2)/2)
    return img[padh:padh+height2, padv:padv+width2]

image = power2crop(image)
print(f'The image is now {image.shape[0]} x {image.shape[1]}')

# plot image
plot.figure(figsize=(20,10))
plot.title('The image that we will use')
plot.imshow(image)
plot.show()
```

```
The image is now 2048 x 4096
```

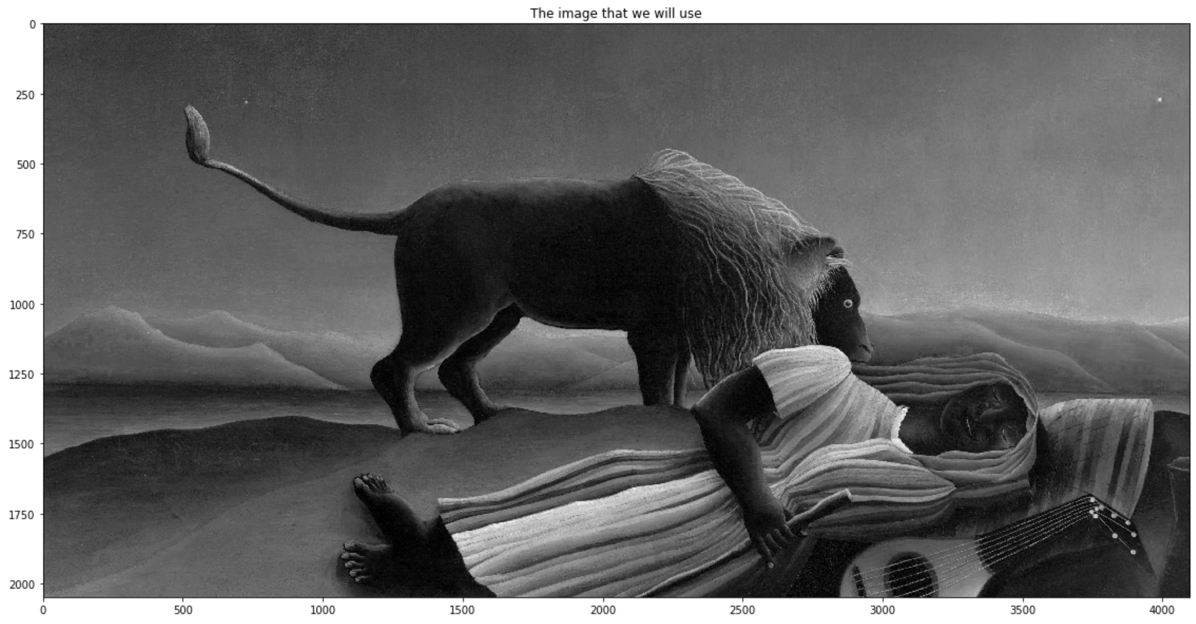
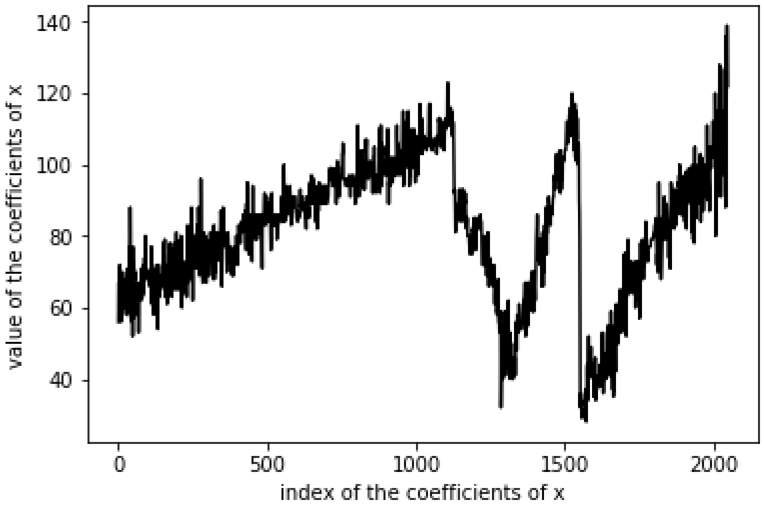


Image compression

We will use the Haar wavelet basis to compress our image. We will see each column of pixels as a vector in $\mathbb{R}^{2048}$, and compute their coordinates in the Haar wavelet basis of $\mathbb{R}^{2048}$.

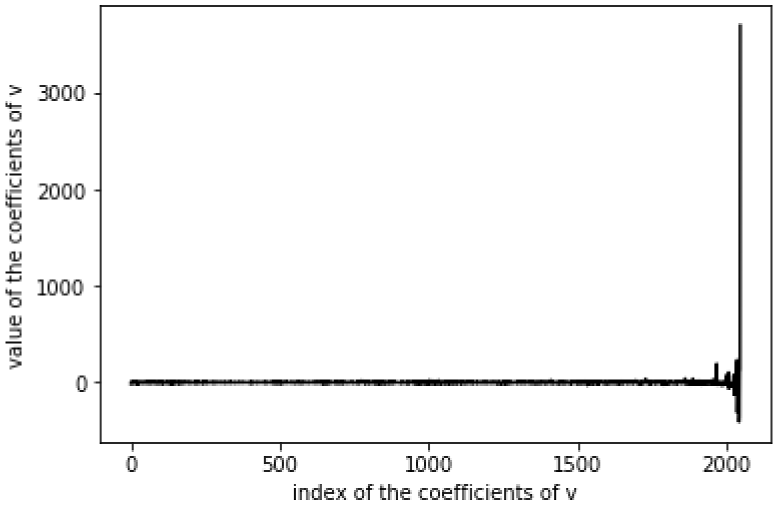
In [55]:
```python
H = haar(11) # matrix of the Haar basis of dimension 2**11 = 2048
x = image[:,0]

# Plot the entries of x, the first column of our image
plot.plot(x,color='black')
plot.xlabel('index of the coefficients of x')
plot.ylabel('value of the coefficients of x')
plot.show()
```



(c) Compute the vector v of the coordinates of x in the Haar basis, and plot its entries.

Explain intuitively why there are only few coefficients of significant magnitude.

In [56]:
```python
v = np.dot(H.T, x)
plot.plot(v,color='black')
plot.xlabel('index of the coefficients of v')
plot.ylabel('value of the coefficients of v')
plot.show()
```



(d) Compute the 2048 x 4096 matrix `wavelet_coeffs` whose columns are the wavelet coefficients of the columns of `image` .

In [57]:
```
wavelet_coeffs = np.dot(H.T, image)
wavelet_coeffs.shape
```

Out[57]:
```
(2048, 4096)
```

Since a large fraction of the wavelet coefficients seems to be negligible, we see that the vector `v` can be well approximated by a linear combination of a small number of wavelets.

Hence, we can 'compress' the image by only storing a few wavelet coefficients of largest magnitude.

Let say that we want to reduce the size by 97%: Store only the top 3% largest (in absolute value) coefficients of `wavelet_coeffs.

(e) Compute a matrix `thres_coeffs` who is the matrix `wavelet_coeffs` where about 97% smallest entries have been put to 0.

In [58]:
```
vals = wavelet_coeffs.flatten()
print(max(vals))
print(min(vals))
thresh = np.percentile(np.absolute(vals), 97)
print(thresh)
thres_coeffs = wavelet_coeffs
thres_coeffs[(-1*thresh < thres_coeffs) & (thres_coeffs < thresh)] = 0
```

```
4976.993176468426
-1308.625
41.01219330881975
```

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

In [59]:
```
compressed_image = np.dot(H, thres_coeffs)

plot.figure(figsize=(20,10))
plot.title('Compressed Image')
plot.imshow(compressed_image)
plot.show()
```

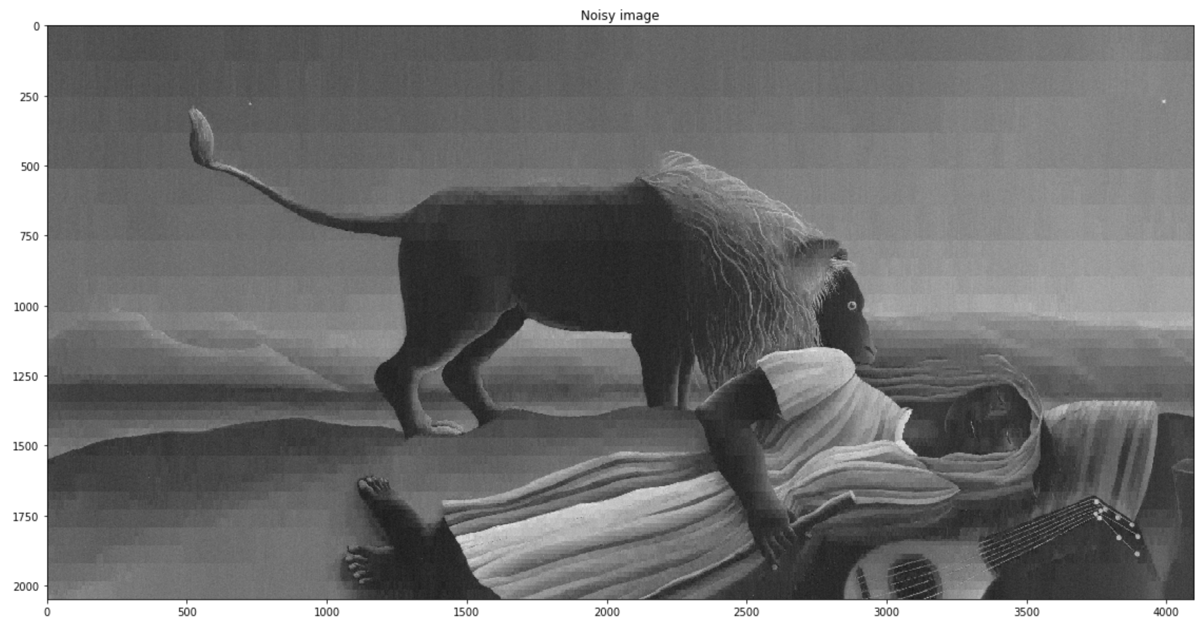
Compressed Image
0
250
500
750
1000
1250
1500
1750
2000
0
500
1000
1500
2000
2500
3000
3500
4000

3. Image denoising

From now on, we will work with the `compressed_image` computed in the previous section, for which about 97% of wavelet coefficients are 0. (We make this choice only in order to make the analysis easier).

We suppose that our `compressed_image` has been corrupted by some Gaussian noise of variance σ:

In [60]:
```
sigma=5
Z=np.random.normal(size=compressed_image.shape) # matrix filled with independe
nt standard Gaussian random variables

noisy_image = compressed_image + sigma * Z

plot.figure(figsize=(20,10))
plot.imshow(noisy_image)
plot.title('Noisy image')
plot.show()
```



Theoretical analysis

We would like to denoise the `noisy_image`. Again, we will denoise each column separately.

Let x be a column of the `compressed_image` and y the corresponding column of `noisy_image`. Then we have

$$y_j = x_j + \sigma z_j$$

where the z_j are iid $\mathcal{N}(0, 1)$.

Without knowledge of the structure of x, one can not really improve on the stupid estimator $\hat{x}_{\text{stupid}} = y$ that achieve and error

$$\|x - \hat{x}_{\text{stupid}}\|^2 = \|\sigma z\|^2 \simeq \sigma^2 N,$$

where $N = 2048$ is the height of the image. We see here that the error grows linearly in N, which is an issue, when N is very large.

However, we know from the previous section that x is 'sparse in the wavelet basis', ie **a large fraction of the wavelet coefficients of x are 0**. Only s wavelet coefficients of x are non-zero (here, we have $s \simeq 0.03 \times N$).

Using this information we will construct in the following an estimator $\hat{x}$ for which

$$\|x - \hat{x}\|^2 \leq 8\sigma^2 s \log(N),$$

In many situation, $s << N$, hence this denoising procedure leads to a huge improvement.

The key is again to study the problem in the Haar wavelet basis. Let v and w be the coefficients of respectively x and y in the Haar wavelet basis. **Optional**: Justify that

$$w_j = v_j + \sigma \epsilon_j$$

where the ϵ_j are iid $\mathcal{N}(0, 1)$.

Probability theory tells us that for large N,

$$\max \epsilon_i \simeq \sqrt{2 \log(N)},$$

hence we will assume in the sequel that $|\epsilon_i| \leq \sqrt{2 \log(N)}$ for all i. It has been proposed to estimate v from w by "soft-thresholding" the coefficient of v: $$ \hat{v}_j =

\eta\Big(w_j, \sigma \sqrt{2 \log N}\Big)

$$\begin{cases} 0 & \text{if } |w_j| \leq \sigma\sqrt{2\log N} \\ w_j - \sigma\sqrt{2\log N} & \text{if } w_j \geq \sigma\sqrt{2\log N} \\ w_j + \sigma\sqrt{2\log N} & \text{if } w_j \leq -\sigma\sqrt{2\log N} \end{cases}$$

$$ (we basically set the coefficients smaller than $\sigma \sqrt{2 \log N}$ to zero, but in order to avoid creating discontinuities, we also shrink the larger coefficients).

(g) Give an intuitive motivation for the procedure above.

---------ANSWER----------

This method is reasonable, because if $|w_j| \leq \sigma\sqrt{2\log N}$, then we know that v_j was already not very significant, since v_j could be at most $\sigma\sqrt{2\log N}$. Then, since v_j is relatively small and insignificant, we set it equal to zero as we would have done anyways, because it is surely not within the top 3% of data values. The other aspects of $\hat{v}_j$ are simply to avoid large discontinuities, and push maximal values closer together.

---------ANSWER----------

(h) Justify that

$$\|v-\hat{v}\|^2 \leq 8\sigma^2 s\log(N),$$

and deduce and estimator $\hat{x}$ such that

$$\|x-\hat{x}\|^2 \leq 8\sigma^2 s\log(N).$$

---------ANSWER----------

We wish to find the maximum error that could exist, given by

$$||v-\hat{v}||^2$$

So we start by considering v_j. The maximum possible error associated with $||v_j-\hat{v}_j||^2$ occurs when

$$w_j = v_j + \sigma\epsilon_j = v_j + \sigma\sqrt{2\log(N)}$$

Meaning that,

$$v_j = w_j - \sigma\sqrt{2\log(N)}$$

Then we know that if soft-thresholding yields

$$\hat{v}_j = w_j + \sigma\sqrt{2\log N}$$
$$v_j - \hat{v}_j = -2\sigma\sqrt{2\log N}$$

We get,

$$||v_j-\hat{v}_j||^2 = 8\sigma^2\log N$$

Furthermore, for $||v-\hat{v}||^2$, we know that at most only $0.03N$ of values could satisfy this aspect of soft-thresholding, since the others are too small and will be set to zero. So, in this case we would have,

$$||v-\hat{v}||^2 = 8\sigma^2 s\log N$$

Since $s = 0.03N$. So, in general then,

$$||v-\hat{v}||^2 \leq 8\sigma^2 s\log N$$

Furthermore, we know that the transition between x and v is governed by the matrix associated by the Haar basis. But applying an orthogonal matrix to some vector does not change that vector's magnitude. Therefore, this inequality will also hold as follows,

$$||x-\hat{x}||^2 \leq 8\sigma^2 s\log N$$

---------ANSWER----------

Application

(i) Apply the method studied above to denoise the `noisy_image` . You have to compute and plot a matrix `denoised_image` .

We give below the soft-thresholding function η:

```
In [61]: def soft_thresholding(data,value):
             return data/np.abs(data) * np.maximum(np.abs(data) - value, 0)
```

```
In [62]: data = np.dot(H.T,noisy_image)
         corrected = soft_thresholding(data,(5*np.log(2048))**0.5)
         denoised_image = np.dot(H,corrected)
```

We check below that the `denoised_image` is closer from `compressed_image` than `noisy_image` :

```
In [63]: def square_error(img1,img2):
             h,w = img1.shape
             return np.sum(np.square(img1-img2)) /(h*w)

         error_naive = square_error(noisy_image, compressed_image)
         error_denoised = square_error(denoised_image, compressed_image)

         print(f'Error of the naive estimator: {error_naive} (this should be close to s
         igma**2)')
         print(f'Error of the denoised image: {error_denoised}')
```

```
Error of the naive estimator: 24.990565186143836 (this should be close to sig
ma**2)
Error of the denoised image: 4.023196197108329
```

Concluding remarks

You should observe that the error achieved by the `denoised_image` is about the half of the one of the naive estimator.

This is very nice. In practice, one can do much better:

- There exists **better thresholds** than $\sigma\sqrt{2\log(N)}$, that leads to a smaller error.
- In practice, we do not perform the wavelet decomposition over the columns (of size 2048) but over the all image using **'2D' wavelets**. Instead of having $N = 2048$ we have now $N = 2048 \times 4096$, hence the improvement from $\sigma^2 N$ to $8\sigma^2 s \log(N)$ is much bigger.
- Using 2D-wavelets would also remove many of the 'horizontal edges' that we see on the compressed and denoised images.
- Here, we used for simplicity the Haar wavelets, but there exists **other wavelets** that are better for image compression/denoising. By 'better', we mean 'sparser images wavelet coefficients': the sparser v is, the smaller s is in the bound $\leq 8\sigma^2 s \log(N)$.
- Ultimately, it is possible to **'learn' a good basis** in which images are sparse, see for instance this tutorial (http://lear.inrialpes.fr/people/mairal/tutorial_cvpr2010/).