3. Norm and distance

3.1. Norm

Norm. The norm $\|x\|$ can be computed in Python using `np.linalg.norm(x)`. (It can be evaluated in several other ways too.) The `np.linalg.norm` function is contained in the `numpy` package `linalg`.

```
In [ ]: x = np.array([2,-1,2])
        print(np.linalg.norm(x))
```

```
3.0
```

```
In [ ]: print(np.sqrt(np.inner(x,x)))
```

```
3.0
```

```
In [ ]: print((sum(x**2)**0.5)
```

```
3.0
```

Triangle inequality. Let's check the triangle inequality, $\|x + y\| \leq \|x\| + \|y\|$, for some specific values of x and y.

```
In [ ]: x = np.random.random(10)
        y = np.random.random(10)
        LHS = np.linalg.norm(x+y)
        RHS = np.linalg.norm(x) + np.linalg.norm(y)
        print('LHS:', LHS)
        print('RHS:', RHS)
```

```
LHS: 3.6110533105675784
RHS: 3.8023691306447676
```

Here we can see that the right-hand side is larger than the left-hand side.

RMS value. The RMS value of a vector x is $\mathbf{rms}(x) = \|x\|/\sqrt{n}$. In Python, this is expressed as `np.linalg.norm(x)/np.sqrt(len(x))`. Let's define a vector (which represents a signal, *i.e.* the value of some quantity at uniformly space time instances), and find its RMS value.

```
In [ ]: rms = lambda x: (sum(x**2)**0.5)/(len(x)**0.5)
        t = np.arange(0,1.01,0.01)
        x = np.cos(8*t) - 2*np.sin(11*t)
        print(sum(x)/len(x))
```

```
-0.04252943783238685
```

```
In [ ]: print(rms(x))
```

```
1.0837556422598
```

```
In [ ]: import matplotlib.pyplot as plt
        plt.ion()
        plt.plot(t,x)
        plt.plot(t, np.mean(x)*np.ones(len(x)))
        plt.plot(t, (np.mean(x) + rms(x))*np.ones(len(x)), 'g')
        plt.plot(t, (np.mean(x) - rms(x))*np.ones(len(x)), 'g')
        plt.show()
```

The above code plots the signal, its average value, and two constant signals at $\mathbf{avg}(x) \pm \mathbf{rms}(x)$ (Figure 3.1).

Chebyshev inequality. The Chebyshev inequality states that the number of entries of an n-vector x that have absolute value at least a is no more than $\|x\|^2/a^2 = n\mathbf{rms}(x)^2/a^2$. If the number is, say, 12.15, we can conclude that no more than 12 entries have absolute value at least a, since the number of entries is an integer. So the Chebyshev bound can be improved to be $floor(\|x\|^2/a)$, where $floor(u)$ is the integer part of a positive number. Let's define a function with the Chebyshev bound, including the floor function improvement, and apply the bound to the signal found above, for a specific value of a.

```
In [ ]: # Define Chebyshev bound function
        import math
        cheb_bound = lambda x,a: math.floor(sum(x**2)/a)
        a = 1.5
        print(cheb_bound(x,a))
```

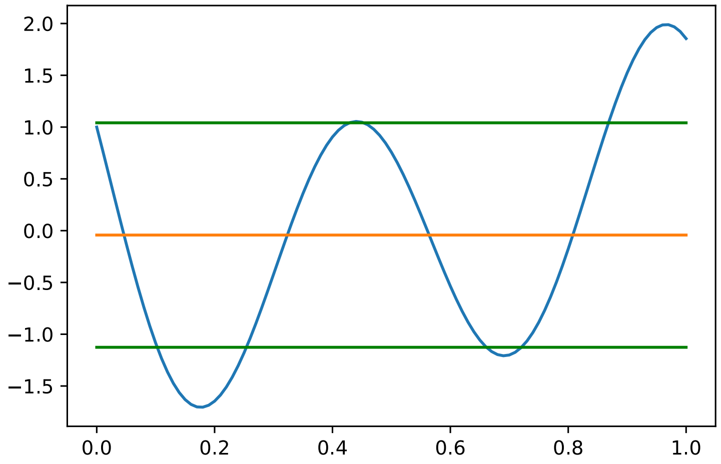

Figure 3.1.: A signal x. The horizontal lines show $\mathbf{avg}(x) + \mathbf{rms}(x)$, $\mathbf{avg}(x)$, and $\mathbf{avg}(x) - \mathbf{rms}(x)$.

```
79
```

In [ ]:
```
# Number of entries of x with |x_i| >= a
print(sum(abs(x) >= a))
```

```
20
```

In the last line, the expression `abs(x) >= a` creates an array with entries that are Boolean, *i.e.*, `true` or `false`, depending on whether the corresponding entry of `x` satisfies the inequality. When we sum the vector of Boolean, they are automatically converted to the numbers 1 and 0, respectively.

3.2. Distance

Distance. The distance between two vectors is $\mathbf{dist}(x, y) = \|x - y\|$. This is written in Python as `np.linalg.norm(x-y)`. Let's find the distance between the pairs of the three vectors u, v, and w from page 49 of VMLS.

In [ ]:
```
u = np.array([1.8, 2.0, -3.7, 4.7])
v = np.array([0.6, 2.1, 1.9, -1.4])
w = np.array([2.0, 1.9, -4.0, 4.6])
```