

Wavelets MATLAB Assignment 5

5.1: Linear Convolution in MATLAB

We noticed back in MATLAB assignment 2, section 2, that the MATLAB command “conv” does not perform circular convolution. Actually it performs linear convolution, that is, convolution in the sense of vectors in $\ell^2(\mathbb{Z})$, at least for two vectors with only finitely many non-zero components. (No program can compute infinite sums exactly.) For example, try

```
>> z=[2 4 6]
>> w=[1 5]
>> conv(z,w)
```

You should get the vector [2 14 26 30]. Notice that we got an answer even though z and w have different lengths, which is not possible with circular convolution. What does the answer for $\text{conv}(z,w)$ above mean? If you think of z as the vector in $\ell^2(\mathbb{Z})$ of the form

$$z = [\dots, 0, 0, 0, 2, 4, 6, 0, 0, 0, \dots]$$

and similarly

$$w = [\dots, 0, 0, 0, 1, 5, 0, 0, 0, \dots],$$

then

$$z * w = [\dots, 0, 0, 0, 2, 14, 26, 30, 0, 0, 0, \dots]$$

(where convolution is defined as in Definition 4.32 in the text). However, this seems unclear because the indices are not specified. If we define z and w so that $z(0) = 2, z(1) = 4, z(2) = 6, w(0) = 1, w(1) = 5$ (and all other components of z and w are 0), then for example

$$z * w(0) = \sum_{n \in \mathbb{Z}} z(0-n)w(n) = \sum_{n=0}^1 z(-n)w(n) = z(0)w(0) + z(-1)w(1) = 2 \cdot 1 + 0 \cdot 5 = 2,$$

which tells us that 2 is the component $z * w(0)$ of $z * w$.

However, this is not the only interpretation. For example, suppose we define z and w so that $z(7) = 2, z(8) = 4, z(9) = 6, w(12) = 1, w(13) = 5$, and all other components of z and w are 0. Then, for example,

$$z * w(19) = \sum_{n \in \mathbb{Z}} z(19-n)w(n) = \sum_{n=12}^{13} z(19-n)w(n) = z(7)w(12) + z(6)w(13) = 2 \cdot 1 + 0 \cdot 5 = 2,$$

so 2 is the component $z * w(19)$ of $z * w$. How does one get the number 19?

Let's consider this case for general vectors z and w with only finitely many nonzero components. Recall Exercise 4.7.6, which states that if $z(n) = 0$ for $n < a$ and for $n > b$, and $w(n) = 0$ for $n < c$ and for $n > d$, then $z * w(n) = 0$ for $n < a + c$ and for $n > b + d$. MATLAB output can be interpreted in a way that is consistent with this fact. That is, MATLAB represents the infinite vector z which is 0 except for $a \leq n \leq b$ by the finite vector

$$[z(a) \ z(a+1) \ z(a+2) \ \cdots \ z(b)]$$

without losing information because all other components are 0, and similarly MATLAB represents w by the finite vector

$$[w(c) \ w(c+1) \ w(c+2) \ \cdots \ w(d)].$$

Then the MATLAB output vector should be interpreted as

$$[z * w(a+c) \ z * w(a+c+1) \ z * w(a+c+2) \ \cdots \ z * w(b+d)],$$

which contains all necessary information about $z * w$ since all other components of $z * w$ must be 0 by Exercise 4.7.6. This does not mean that we cannot start a vector with 0 if we choose to do so; if for example we let

```
>> z=[ 2 4 6]
>> w=[ 0 1 5]
```

then

```
>> conv(z,w)
```

gives the answer [0 2 14 26 30]. This reflects the fact that if we let $z(a) = 2, z(a+1) = 4, z(a+2) = 6, w(c) = 0, w(c+1) = 1,$ and $w(c+2) = 5$ then $z * w(a+c) = 0$. What is most important is that the MATLAB output vector should be regarded as the components of $z * w$ starting with index $a+c$ and ending with index $b+d$, where the displayed components for z have indices going from a to b and for w going from c to d . Note that the number of displayed components for $z * w$ is always the number displayed for z plus the number for w minus 1.

It will be important to keep track of the indices associated to the components of a convolution for two reasons: to know the locations at which to plot the wavelet coefficients, and to know which components to delete when downsampling.

5.2: Daubechies's Filters for \mathbb{Z}

The non-zero components of Daubechies's D6 wavelet filters u and v for \mathbb{Z} are the same as for u_1 and v_1 in the case of \mathbb{Z}_N that we considered back in Section 5 of MATLAB assignment 2. One just has to be careful to get them in the correct order and to interpret their indices correctly.

Let's start by loading the vectors $u_1, v_1, \tilde{u}_1,$ and \tilde{v}_1 . In Section 5 of MATLAB assignment 2 we called these "dauu1", "dauv1", "dauu1til" and "dauv1til" and saved these in a file called "dauwavefilt".

```
>> load dauwavefilt
```

This yields (among other vectors) the vector `u1` of length 512, whose only non-zero coefficients are the first 6 (namely `u1(0), u1(1), u1(2), u1(3), u1(4), u1(5)`). By comparing Example 3.35 with Example 4.57, we see that the infinite vector u for the case of \mathbb{Z} is 0 except for the components with indices 0, 1, 2, 3, 4, and 5, and the values for these indices are the same as for `u1`. In other words, we can represent u via the finite vector `[u(0), u(1), u(2), u(3), u(4), u(5)]` (with the usual understanding that all other components are 0) via

```
>> u = dauu1(1:6)
```

(not `(0:5)` because MATLAB regards the first component as `u1(1)` not `u1(0)`).

Then, as described in Example 4.57, the nonzero components of v are $v(-4) = -u(5), v(-3)=u(4), v(-2)=-u(3), v(-1)=u(2), v(0) = -u(1),$ and $v(1)=u(0)$. Because of the way MATLAB views the indices, to get a representation of v one could type

```
>> v = [ -u(6)  u(5)  -u(4)  u(3)  -u(2)  u(1) ]
```

or, considering the vector `v1` stored from before, one can use

```
>> v = [dauv1(509:512)  dauv1(1:2)]
```

to get the same thing.

However, the filters used in computing the (forward) wavelet transform are the tildes of u and v . Recall that $\tilde{z}(n) = z(-n)$. Here u and v are real, so we can ignore the complex conjugate, but we do have to be careful about the indices for \tilde{u} and \tilde{v} . For example, since $u(n) \neq 0$ only for $0 \leq n \leq 5$, we will have $\tilde{u}(n) \neq 0$ only for $-5 \leq n \leq 0$. More specifically, we will have $\tilde{u}(-5) = u(5), \tilde{u}(-4) = u(4), \tilde{u}(-3) = u(3), \tilde{u}(-2) = u(2), \tilde{u}(-1) = u(1), \tilde{u}(0) = u(0),$ and all other components are 0. Thus we represent \tilde{u} , which we will call "util" in MATLAB, via

```
>> util = [ u(6)  u(5)  u(4)  u(3)  u(2)  u(1) ]
```

(again, the indexing is changed to reflect MATLAB's convention), where we have to recall that the indices of the non-zero components of "util" are -5 to 0. Similarly, $v(n) \neq 0$ only for $-4 \leq n \leq 1$, so $\tilde{v}(n) \neq 0$ only for $-1 \leq n \leq 4$. We obtain the MATLAB representation "vtil" by

```
>> vtil = [v(6)  v(5)  v(4)  v(3)  v(2)  v(1) ]
```

but we need to keep in mind that the indices with non-zero components of "vtil" are -1 to 4.

At this point it is worth saving the filters $u, v, \text{util},$ and vtil in a file with a name you can distinguish from the corresponding name for the filters on \mathbb{Z}_{512} earlier; I suggest a filename "dauzfil" where the "z" suggest filters on \mathbb{Z} . Notice that we do not need to load different filters corresponding to different levels, like we needed to do in section 2 of MATLAB assignment 3, since in \mathbb{Z} we can use the same filters at all levels.

5.3: "Odd Downsampling"

The first stage wavelet coefficients of a vector z are of the form $D(z * \tilde{u})$ and $D(z * \tilde{v})$. So it may seem like a simple matter to use our downsampling function "down" from Section 4 of MATLAB assignment 2 to compute the wavelet coefficients. After all, "down" works on vectors of any length. However, there is a problem. Define a vector z by $z(0) = 2, z(1) = 4, z(2) = 6$ and all other components are 0:

```
>> z=[2 4 6]
```

Then compute $z * \tilde{v}$:

```
>> y = conv(z, vtil)
```

You should get

$$[.6653 \quad -0.2831 \quad -0.3118 \quad -2.7318 \quad 3.1284 \quad 0.3978 \quad -0.6536 \quad -0.2114]$$

Now try to downsample:

```
>> down(y)
```

You should get

$$[0.6653 \quad -0.3118 \quad 3.1284 \quad -0.6536]$$

Is this correct? Recall that the indices of `vtil` with nonzero components are -1 to 4, while by definition the indices of z with nonzero components are 0 to 2. By the section on convolution above, the indices of the components of $z * \tilde{v}$ displayed above are -1 to 6. When we downsample, we want to delete the components with odd indices, namely with indices -1, 1, 3, and 5, leaving

$$[-0.2831 \quad -2.7318 \quad 0.3978 \quad -0.2114]$$

whereas “down” eliminated the other 4 values. That’s because when we wrote “down”, we assumed all vectors started with index 0, which is even and so the first component should be retained.

One way around this is to write a new downsampling function file that applies to vectors whose first index is odd, which we call “odddown” for “odd downsampling”. This is most easily done by altering the file “down.m” which had the commands

```
n=length(z)/2;
for k=1:n
y(k)=z(2.*k-1);
end
```

and altering the line

```
y(k)=z(2.*k-1);
```

to read

```
y(k) = z(2.*k);
```

Since MATLAB regards vectors as starting with index 1, this defines the first component of y to be the second component of z , and so on, which is what we want since we want to delete the first displayed component. So let’s create such a function file. Don’t forget to rename the function “odddown” on the first line.

5.4: The Wavelet Transform on \mathbb{Z}

Let z be a vector in $\ell^2(\mathbb{Z})$. We will suppose that $z(n) = 0$ for $n < 0$ and for $n > 511$ (for example the vectors z for Figures 39 and 41 required for this assignment). Thus the infinite vector z is represented in MATLAB by the finite vector

$$[z(0), z(1), z(2), \dots, z(511)].$$

We are interested in computing the 4th level wavelet transform of z , namely the vectors x_1, x_2, x_3, x_4 and y_4 . Recall that

$$x_1(k) = \langle z, R_{2k}v \rangle = z * \tilde{v}(2k) = D(z * \tilde{v})(k),$$

$$y_1(k) = \langle z, R_{2k}u \rangle = z * \tilde{u}(2k) = D(z * \tilde{u})(k).$$

Recall that \tilde{v} is a vector with 6 nonzero components $\tilde{v}(-1), \tilde{v}(0), \dots, \tilde{v}(4)$, which we represent by a vector of length 6 in MATLAB. Therefore when we compute

```
>> y = conv(z, vtil)
```

we will get a vector of length $512 + 6 - 1 = 517$, whose components correspond to the indices going from -1 to 515 (since $0 + (-1) = -1$, whereas $511 + 4 = 515$). Since the first index of the displayed components is odd, we need to use “odddown” from the previous section to downsample:

```
>> x1=odddown(y)
```

The even numbers between -1 and 515 go from 0 to 514, so for x1 you should get a vector of length 258, corresponding to $x_1(0), x_1(1), \dots, x_1(257)$. Since $x_1(k) = \langle z, R_{2k}v \rangle$, we want to plot the point $x_1(k)$ at the location $2k$. So we could define

```
>> m1=0:2:514;
```

Then if we use

```
>> plot(m1,x1, '.')
```

we would get a plot of all the nonzero components of the first level wavelet coefficients of z . However, it is important to realize that $x_1(k)$ is defined for all $k \in \mathbb{Z}$; it has the value 0 for all other k . To exhibit this, it is natural to include a few of the 0 values at the right and left of the picture. So for example in Figure 39b, the limits on the horizontal axis go from -10 to 534. So instead we can use

```
>> n1=-10:2:534;
```

But now we cannot use “plot(n1,x1)” since x1 has only 258 components while n1 has 273 components. We need to put in a 0 for the components with indices -10, -8, -6, -4, -2, 516, 518, 520, 522, 524, 526, 528, 530, 532, and 534. The easiest way to do this is just to define

```
>> x1pl= [zeros(1,5) x1 zeros(1,10)];
```

where “x1pl” stands for “x1plot”. Now

```
>> plot(n1, x1pl, '.')
```

should give the right picture (after resizing using the axis feature).

Similarly, to compute y_1 , recall that \tilde{u} has 6 non-zero components with indices running from -5 to 0. Therefore the indices for the components of conv(z, util) in MATLAB will run from -5 to 511. Therefore we should use “odddown” again for downsampling:

```
>> y1=odddown(conv(z, util));
```

The even indices between -5 and 511 run from -4 to 510. Since $y_1(k) = z * \tilde{u}(2k)$, the vector y1 will have 258 components with indices running from -2 to 255. This is important to keep in mind at the next step.

Now recall that $x_2 = D(y_1 * \tilde{v})$. In MATLAB, the displayed components of y1 have indices that run from -2 to 255, while the indices of the displayed components of vtil run from -1 to 4. So the indices of the displayed components of

```
>> y = conv(y1, vtil)
```

run from -3 to 259. After deleting the odd ones, the indices run from -2 to 258. Therefore

```
>> x2=odddown(y)
```

is a vector with 131 displayed components, with indices going from -1 to 129. These should be plotted at the points $4k$, as k runs from -1 to 129 (along with as many zeros to the left and right of these values as one wishes to put in).

Similarly $y_2 = D(y_1 * \tilde{u})$. Since the indices for the nonzero components of \tilde{u} run from -5 to 0, the indices for the MATLAB display of “conv(y1, util)” run from -7 to 255. (Recall that the displayed components of y1 have indices running from -2 to 255.) After deleting the odd ones, the even indices go from -6 to 254. Thus the indices for the MATLAB display of “odddown(conv(y1,util))” should be -3 to 127.

Continuing in this way (the details are for you to work out), one computes x_3, y_3, x_4 , and y_4 . As a warning, one does not always use “odddown”. In at least one case the initial component has an even index, so one should use “down” instead.

Assignment 5

Reproduce the following figures from the text. With each figure, please print out the MATLAB code you used in creating that figure.

1.) Figures 39b, 39d, and 39f. The limits on the vertical axes can be read off from the figures. The limits on the horizontal axes are as follows: Figure 39b: -10 to 534; Figure 39d: -40 to 544; Figure 39f: -96 to 544.

2.) Figures 41c and 41e. The vertical axes have limits -.6 to .6 in Figure 41c, and -1 to 1 in Figure 41e. The horizontal axes have limits -80 to 528 in both Figures.

Figures 41c and 41e should be compared with Figures 28c and 28e, to see how the periodicity effects have been eliminated by using wavelets on \mathbb{Z} . Reproduce Figures 28c and 28e and compare with 41c and 41e.

3.) The above graphs used D6 filters. Since we also developed D2 and D4 filters, recreate similar graphs with D2 and D4 filters.

Remark: The interested student is encouraged to produce filters DN with $N > 6$, the link for the coefficients is on the class webpage.