Throughout the supplements, we’ve run into ‘edge effects’, either implicitly or explicitly. The issue is simple to state: our linear, time-invariant filters all perform moving averages against a signal. Data signals, however, are always of finite duration, hence the moving average lacks sufficient data at the beginning and end of a dataset.

The text suggests we use zero padding. Let’s try that, with a two-point moving average, and the following padded data:

\[ s = 0 \ 0 \ 8 \ 6 \ 10 \ 8 \ 6 \ 0 \ 0 \]
\[ av = 0 \ 4 \ 7 \ 8 \ 9 \ 7 \ 3 \]

The value of the average is unrealistically low at the two ends. Averaging, of course, is a low-frequency, smoothing filter. The other part of the time-frequency analyses we perform is a differencing, or high frequency filter. as this function like a derivative, the edge effects are much worse. Recall that our generic time-frequency filter computes a derivative. The zero-padded signal \( s \) above has an enormous derivative at the edge. The effect is visible in most of the examples we’ve examined:

Top: heart rate signal (measured in beats per minute). Bottom: the CWT. The purples and reds at the far ends represent high-energy effects from the edge effects.
Top: Two sine waves, with a discontinuity in the derivative, two-thirds through. Bottom: the CWT. Note the effects due to the discontinuity at the edge are comparable to the discontinuity we’re seeking to locate two-thirds though. This example shows how edge effects may confound the features we want.

Left: Vacation picture, English Bay, Vancouver.
Right: Wavelet transform, employed to detect presence of the bird against the background. Again: the feature we want to extract is the bird, not the edge of the picture.
The bird example is perfect for our purpose, because JPEG2000 in fact uses wavelets to compress images. The compressed image had better not have edge effects! How is this accomplished? Let’s find out.

We’ll take an example from the exciting world of biomedical consulting. A client, whose identity is confidential, needs information on the effects of certain “interventions”. You know heart rate will be recorded. You are not told what the interventions consist of; you are merely asked to identify when they occur, as accurately as you can, using wavelet technology.

As mentioned before, the heart is regulated by sympathetic and parasympathetic mechanisms, and the effect of each of these can be detected in the frequencies constituting the BPM signal. Amplitudes from .0625Hz to .1875Hz are ‘low-frequency’ for these purposes, while the high frequency range is .1875 to .4375Hz.

We’ll use the wavelet packet transform to access these frequencies. Our BPM data is sampled at 1Hz, so that the Nyquist range is [0, .5] Hz. A level one decomposition divides this into two regions of length .25Hz; a level two into 4 regions of length .125Hz; a level three into 8 regions of length .0625Hz. This is the correct frequency resolution for us.

The client specifies that it wants the highest possible time resolution; this suggests we take the shortest length wavelets available, the Daubechies 2 family. Given these two requirements, the Matlab code follows:

```matlab
function modeplot(s, mode)

dwtmode(mode);
wpt = wpdec(s,3,'db2');
lo1=wprcoef(wpt, [3 1]);
lo2=wprcoef(wpt, [3 3]);
hi1=wprcoef(wpt, [3 2]);
hi2=wprcoef(wpt, [3 6]);
hi3=wprcoef(wpt, [3 7]);
hi4=wprcoef(wpt, [3 5]);
lo0=wprcoef(wpt, [3 0]);
hi5=wprcoef(wpt, [3 4]);
lo=abs(lo1).^2+abs(lo2).^2;
hi=abs((hi1)).^2+abs((hi2)).^2+ abs((hi3)).^2 +abs((hi4)).^2;
subplot(3,1, 1), plot(s), title(' signal')
subplot(3,1, 2), plot(lo), title(' low frequency')
subplot(3,1, 3), plot(hi), title(' high frequency')
```

We haven’t met the phrase “dwtmode(mode);” before; it tells Matlab what method to use in order to deal with edge effects:
Dealing with Border Distortion

Classically, the DWT is defined for sequences with length of some power of 2, and different ways of extending samples of other sizes are needed. Methods for extending the signal include zero-padding, smooth padding, periodic extension, and boundary value replication (symmetrization).

The basic algorithm for the DWT is not limited to dyadic length and is based on a simple scheme: convolution and downsampling. As usual, when a convolution is performed on finite-length signals, border distortions arise.

Signal Extensions: Zero-Padding, Symmetrization, and Smooth Padding

To deal with border distortions, the border should be treated differently from the other parts of the signal.

Various methods are available to deal with this problem, referred to as “wavelets on the interval” (see [CohDJV93] in “References” on page 6-151). These interesting constructions are effective in theory but are not entirely satisfactory from a practical viewpoint.

Often it is preferable to use simple schemes based on signal extension on the boundaries. This involves the computation of a few extra coefficients at each stage of the decomposition process to get a perfect reconstruction. It should be noted that extension is needed at each stage of the decomposition process.

Details on the rationale of these schemes can be found in Chapter 8 of the book Wavelets and Filter Banks, by Strang and Nguyen (see [StrN96] in “References” on page 6-151).

The available signal extension modes are as follows (see dwtmode):

- Zero-padding (‘zpd’): This method is used in the version of the DWT given in the previous sections and assumes that the signal is zero outside the original support.
  
  The disadvantage of zero-padding is that discontinuities are artificially created at the border.
As suggested by our text, let’s try zero-padding:

```matlab
>> load 'secret.txt' -ascii
>> modeplot(secret, 'zpd')
```
And here’s the output:

![Graphs of signal, low frequency, and high frequency components]

Signal at the top; high and low frequency components at the bottom. We see the expected corruption of the results by the discontinuities at the edges.

The first difficulty is that the edge effects are so large that they distort the scale of the graph; if there’s anything happening between the edges, we can’t see it. The easy work-around is to simply not graph the first and last 15 or so points. This roughly corresponds to the length of the Daubechies 2 filters at level 3, so we are in essence discarding all data that comes from applying the filter near an edge.

```matlab
>> load 'secret.txt' -ascii
>> length(secret)
ans = 179
```

Now rewrite the code in modeplot.m

```matlab
subplot(3,1, 1), plot(s(20:159)), title('signal')
subplot(3,1, 2), plot(lo(20:159)), title('low frequency')
subplot(3,1, 3), plot(hi(20:159)), title('high frequency')
```
And run the program. Here’s the output:

![Graphs](image)

Interesting! We can even locate where at least one of the interventions might be! (though there seems to be a phase problem between hf and lf). This is one of the difficulties of the conditions imposed. If you knew the nature of the interventions, you could use your knowledge of physiology, and try to determine whether the phase gap was realistic.

On the other hand, if you use too much of your knowledge of physiology, you might see effects where none are present.

In either case, Client replies ‘You may not disregard data.” (we wish we were making this up.)

Our task, your task: which of the methods available in Matlab minimizes the edge effects?