# Endpoint flattening

The digital signal processing (e.g. Discrete Fourier Fransform) assumes that the time domain dataset is periodic and repeats.

Suppose a price series starts at 3200 and toggles and wobbles for 800 data samples and ends at the value 2400. The DFT assumes that the price series starts at zero, suddenly jumps to 3200, goes to 2400, and suddenly jumps back to zero and then repeats. The DFT has to create all sorts of different frequencies in the frequency domain to try to achieve this kind of behavior. These false frequencies, generated to match the jumps and the high average price, mask the amplitudes of the true frequencies and make them look like noise.

> Fortunately, this effect can be nearly eliminated by a simple technique called **endpoint flattening**.

## Example

The following chart shows an example data series (green) and the de-trended data at the bottom panel (gold) without endpoint flattening:

image-1626255996599.png

The next example shows the same data series now with endpoint flattening applied to the detrended series:

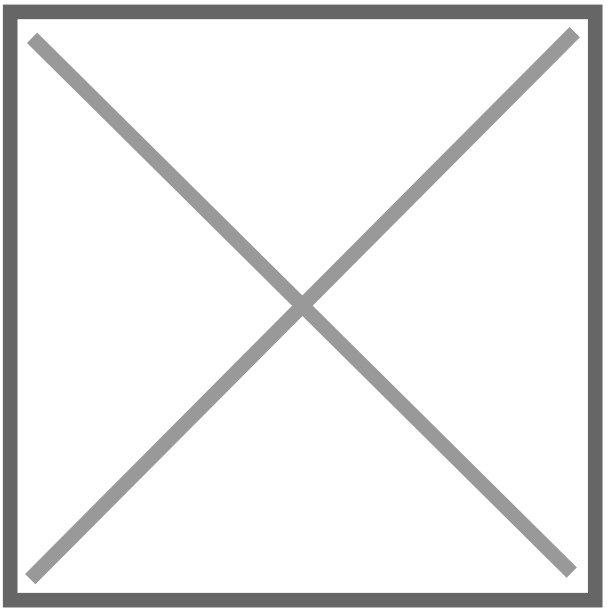
image-1626256103968.png

The difference is only visible at the beginning and the end on both de-trended series. While the first one starts below 0 and ends well above 0, the second chart shows that the de-trended series starts and ends at zero.

## Math formula

Calculating the coefficients for endpoint flattening is simple:

Taking n closing prices. If x(1) represents the first price in the sampled data series, x(n) represents the last point in the data series, and $x_f(i)$ equals the new endpoint flattening series then:



We can see that when i=1 then $x_f(1)=0$ and when i=n then $x_f(n) =0$.

What we've done is subtract the beginning value of the time series to make the first value equal to zero and then rotate the rest of the time series such that the end point is now zero. This technique reduces the endpoint distortion but introduces a low frequency artifact into the Fourier Frequency spectrum. Fortunately we won't be looking for frequencies in that range so this distortion will have minimal impact.

## C# .NET Function example

Apply end-point flattening to an array of double values:

```
public void endpointflattening(double[] values)
{
  int datapoints = values.Count();

  double a = values[0];
  double xn = values[datapoints - 1];
  double b = (xn - a) / (datapoints - 1);

  for (int i = 0; i < datapoints; i++)
  {
  values[i] = values[i] - (a + (b * i));
  }
}
```

Further Reading & References:

- Dennis Meyers Working Papers On Walk-Forward Optimization with Algorithmic Trading Strategies (meyersanalytics.com)

---

Revision #16
Created 12 July 2021 10:26:18 by LvT
Updated 14 July 2021 09:50:46 by LvT