As shown in Figure 8.8 the results from `fft_autocorr` and `np.correlate` are identical (with about 9 digits of precision).

Notice that the correlations in Figure 8.8 are large numbers; we could normalize them (between -1 and 1) as shown in Section 5.6.

The strategy we used here for auto-correlation also works for cross-correlation. Again, you have to prepare the signals by flipping one and padding both, and then you have to trim the invalid parts of the result. This padding and trimming is a nuisance, but that's why libraries like NumPy provide functions to do it for you.

8.8 Exercises

Solutions to these exercises are in `chap08soln.ipynb`.

Exercise 8.1 The notebook for this chapter is `chap08.ipynb`. Read through it and run the code.

It contains an interactive widget that lets you experiment with the parameters of the Gaussian window to see what effect they have on the cutoff frequency.

What goes wrong when you increase the width of the Gaussian, `std`, without increasing the number of elements in the window, `M`?

Exercise 8.2 In this chapter I claimed that the Fourier transform of a Gaussian curve is also a Gaussian curve. For discrete Fourier transforms, this relationship is approximately true.

Try it out for a few examples. What happens to the Fourier transform as you vary `std`?

Exercise 8.3 If you did the exercises in Chapter 3, you saw the effect of the Hamming window, and some of the other windows provided by NumPy, on spectral leakage. We can get some insight into the effect of these windows by looking at their DFTs.

In addition to the Gaussian window we used in this chapter, create a Hamming window with the same size. Zero pad the windows and plot their DFTs. Which window acts as a better low-pass filter? You might find it useful to plot the DFTs on a log-y scale.

Experiment with a few different windows and a few different sizes.