

6 Signal Processing

6.1 Introduction

6.2 Generating Signals

6.3 Linear Time Invariant Systems

Since this system yields equal responses for different input signals, such as $y=4$ for input **signal** $x=-2$ and $x=+2$, the input **signal** x cannot be reconstructed from the output **signal** y .

6.4 Convolution, Deconvolution, and Filtering

6.5 Comparing Functions for Filtering Data Series

Prior to introducing a broader description of linear time invariant filters, we replace `convolve()` with `lfilter()`, which can also be used for *recursive filters*. In this case, the output signal $y(t_n)$ depends not only on the input signal $x(t_n)$, but also on previous elements of the output signal $y(t_{n-1})$, $y(t_{n-2})$, $y(t_{n-3})$, and so on ([Chap. 6.6](#)).

6.6 Recursive and Nonrecursive Filters

6.7 Impulse Response

Furthermore, the values of the output signal y_6 equal the filter weights of **b6**. In our example, these values are 0.2 for all elements of **b6** and y_6 .

The stem plot of the input signal **x7** and the output signal **y7** shows an interesting impulse response:

6.8 Frequency Response

Furthermore, since the magnitude response is the absolute value of the complex frequency response h , **its real part takes on negative values** and lies between ~ 0.09082 and ~ 0.1816 , between ~ 0.2725 and ~ 0.3633 , and between ~ 0.4546 and the Nyquist frequency.

```
plt.figure()
plt.plot(t,x11,color='b')
plt.plot(t,y11,color='r')
plt.axis(np.array([30,40,-2,2]))
plt.show()
```

6.9 Filter Design

We might now wish to design a new filter with a more rapid transition

from passband to stopband. Such a filter requires a higher order, that is, it needs a larger number of filter weights **a13** and **b13**. We can now create a 15-order Butterworth filter as an alternative to the filter above:

6.10 Adaptive Filtering

In the following example, which introduces `canc()`, each of these loop runs is called an iteration, and many loop runs are required if optimal results are to be achieved.

where `kron()` returns the Kronecker tensor product of `yn1` and `yn1'` (which is a matrix formed by taking all possible products between the elements of `yn1` and `yn1'`) and `eig()` returns the **eigenvalue** of `k`.

Recommended Reading

Kalman R, Bucy R (1961) New Results in Linear Filtering and Prediction Theory. ASME **Trans.** Ser. D **J. Basic Eng.** 83:95–107

Figure Captions