Section 8 Convolution and Deconvolution

This document illustrates several techniques for carrying out convolution and deconvolution in Mathcad. There are several operators available for these functions: the summation operator and Mathcad's discrete Fast Fourier Transform functions **dft** and **idft**. You provide:

- **f(t)**, and **g(t),** the two signals to convolve,
- **T0**, the time interval over which the functions are nonzero,
- **N**, the number of sample points.

A quick approximation to the convolution and deconvolution of two continuous functions is performed by sampling, and applying an FFT. Alternately, you can enter:

- two files containing discrete time pulse train amplitudes.

The convolution is performed directly with a summation on the two finite series, and plots of all functions and their convolutions are generated.

References

William McC. Siebert, *Circuits, Signals, and Systems*, The MIT Press (Cambridge, 1986).

Background

Convolution

When a signal passes through a filter, the output is the convolution of the input function with the impulse response of the filter. Convolutions can be tedious if not impossible to perform on complicated signals. Fortunately, the convolution theorem states that convolution in the time domain is equivalent to multiplication in the frequency domain. To compute the convolution of two signals, find their frequency components by Fourier transformation, multiply the components together and find the inverse transform of this product.

The continuous-time Fourier integrals for a nonperiodic function and its transform are given by:

$$x(t) = \int_{-\infty}^{\infty} X(f) \cdot e^{1j \cdot 2 \cdot \pi \cdot f \cdot t} \, df \qquad X(f) = \int_{-\infty}^{\infty} x(t) \cdot e^{-1j \cdot 2 \cdot \pi \cdot f \cdot t} \, dt$$

FFTs

Discrete Fourier transforms (DFTs) are summations rather than integrals. The various symmetries and periodicities of the exponential terms in these summations can be exploited to make the calculation faster, and such a calculation is termed a fast Fourier transform (FFT). For a quick approximation to a continuous-time Fourier transform, we can sample continuous functions **f(t)** and **h(t)** to create the pulse trains **x(n)** and **h(n)**, and then apply FFTs.

Mathcad Implementation

This document shows two ways of carrying out convolutions by Fourier transforming: using the built-in FFT operators, and using the summation operator. The document also illustrates recovery of the original signal by deconvolution.

Approximate Convolution Using FFTs

This example shows a quick approximation to the convolution of two continuous functions by sampling them, multiplying their FFTs and then taking the inverse transform to arrive at the convolved pulse train. The final continuous time solution is then recovered by interpolation. The routine assumes that the functions are only non-zero over a finite time **T0**, and it returns a function **h** giving their convolution over the interval **[**0**,** 2**T0]**.

First, define two continuous functions in time **f** and **g**, and enter for **T0** the longest time for which either function is nonzero, assuming the signals start at time 0. Also, choose a value for **N**, the number of sample points.

Two signals:

$$f(t) := (t \geq 0) \cdot (t \leq 1) \cdot \sin(4 \cdot \pi \cdot t) \qquad H(t) := f(1 - t)$$

Longest time for which either signal is nonzero: $T0 := 1$

Number of sample points: $N := 128$

Next, define a range variable **j** for sampled time and create a vector **n** from the range variable:

$$j := 0 \,..\, N - 1 \qquad n_j := \frac{j}{N} \cdot 2 \cdot T0$$

Using the time samples, construct vectors **x** and **h** with Mathcad's **vectorize** operator, so that they contain the sampled functions:

$$x := \overrightarrow{f(n)} \qquad h := \overrightarrow{H(n)}$$

The convolution of the sampled **f** and **g** is given by:

$$y := \frac{2 \ T0}{\sqrt{N}} \cdot \text{idft}\left(\overrightarrow{\text{dft}(x) \cdot \text{dft}(h)}\right)$$

where the square root of **N** is a normalization factor.

The **linterp** function interpolates linearly between the elements of the discrete convolution, providing an approximately continuous convolution of **f** with **g**.

$$g(t) := \text{linterp}(n, \text{Re}(y), t)$$

$$t := 0, .01 \cdot T0 \,..\, 2 \cdot T0$$

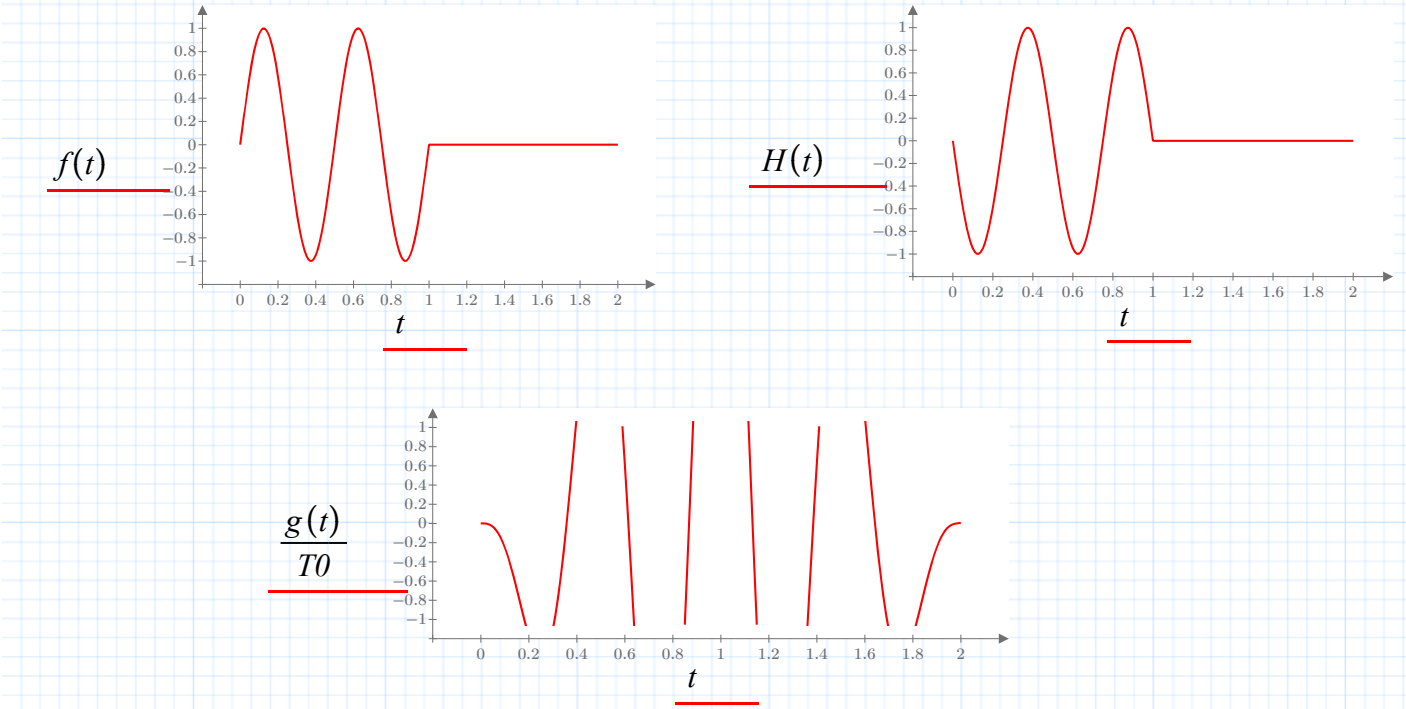

Fig. 8.1 Plots of f(t), H(t) and the approximate continuous time convolution

Approximate Deconvolution Using FFTs

You can carry out deconvolution efficiently by inverting this procedure: divide the FFT of the output sequence by the FFT of one of the inputs. Note that "divide" here means divide each term of one sequence by the corresponding term of the other. In general, some terms in the denominator sequence will be zero, but you can usually avoid a division by zero error and still obtain an accurate deconvolution by adding a small safety factor to each term of the denominator.

$\varepsilon := 10^{-6}$ (a small safety factor)

The deconvolution is given by

$$x' := \frac{\sqrt{N}}{2 \cdot T0} \cdot \mathrm{Re}\left(\mathrm{idft}\left(\frac{\overrightarrow{\mathrm{dft}(y)}}{\mathrm{dft}(h) + \varepsilon}\right)\right)$$

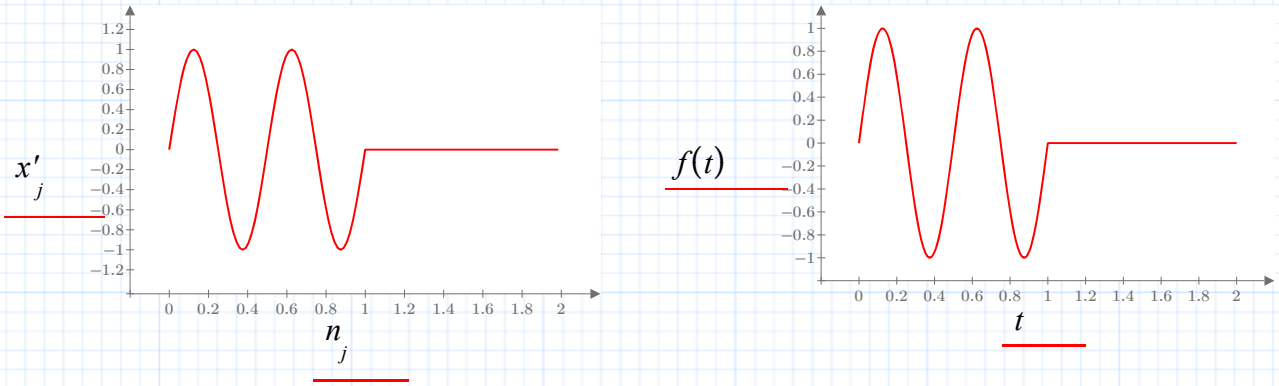

Fig. 8.2 Deconvolved signal compared with original signal

The plot shows that the sequence given by the deconvolution matches the original sampled input function **f**. Notice that the interpolation is not really necessary with this many points, since the graphing function will connect them smoothly anyway.

Discrete Convolution of Two Finite Sequences

This example uses Mathcad's **sum** and **mod** operators to find the convolution of two finite sequences. The arrays **S1** and **S2** are the two sequences to be convolved .

$$S1 := \begin{bmatrix} -0.125 & 0.25 & -0.5 & 1 & -0.5 & 0.25 & -0.125 \end{bmatrix}^{T}$$

$$S2 := \begin{bmatrix} 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 0 \end{bmatrix}^{T}$$

The length of the convolution will be one less than the sum of the lengths of the two sequences. This means that each sequence must have a length of **K**–1.

$M := \text{length}(S1)$ $\quad N := \text{length}(S2)$ $\quad K := M + N$

$m := 0 .. M - 1$ $\quad n := 0 .. N - 1$

Pad each sequence with zeros to the correct length in a new vector:

$X := S1$ $\quad H := S2$

$X_{M+n} := 0$ $\quad H_{N+m} := 0$

$i := 0 .. M + N - 2$ $\quad j := 0 .. M + N - 2$

The convolution is given by

$$C_j := \sum_i \left(X_i \cdot H_{\text{mod}(j - i + K, K)} \right)$$

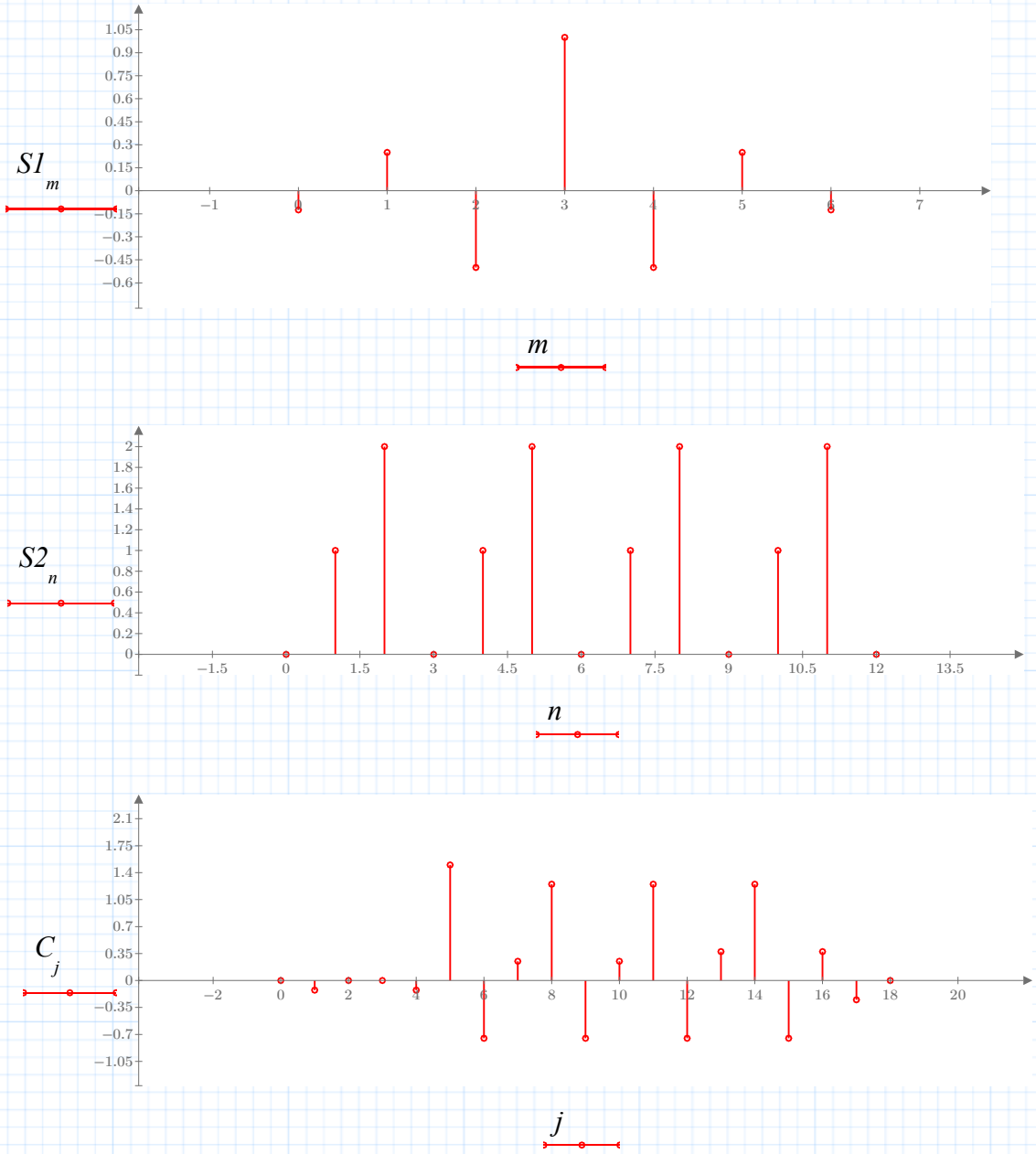

Fig. 8.3 Input pulse trains and their convolution

You can also use the FFT techniques of the first two examples for finite sequences. You will need to pad the sequences to a length equal the sum of their original lengths. Follow this procedure in the last example for a model.