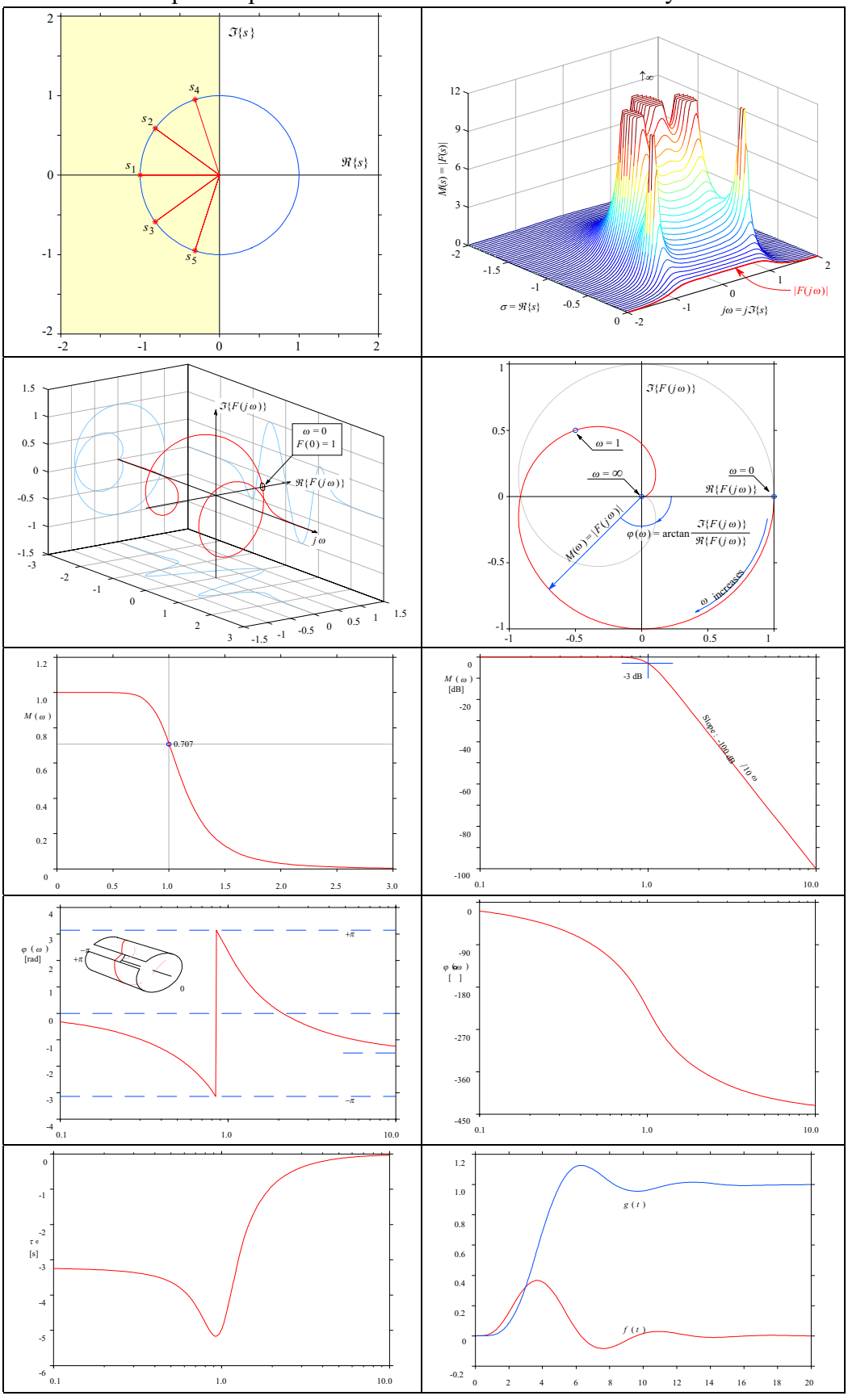
Complete representation of a 5th-order Butterworth system



6.5. Transient Response by Fourier Transform

There are several methods for time-domain response calculation. Three of these, that are interesting from the system designer's point of view, including the FFT method, were compared for efficiency and accuracy in [Ref. 6.6]. Besides the high execution speed, the main advantage of the FFT method is the fact that we don't even have to know the exact mathematical expression for the system frequency response, but only the graph data (i.e. if we have measured the frequency and phase response of a system). Although the method was described in detail in [Ref. 6.6], we will repeat here the most important steps, to allow the reader to follow the algorithm development.

There are **five difficulties,** associated with the discrete Fourier transform, that we will have to solve :

a) the inability to transform some interesting functions (i.e. the unit-step) ;

b) the correct treatment of the d.c. level in low-pass systems ;

c) preserving accuracy with as little spectral information input as possible ;

d) find to what extent our result is an approximation due to finite spectral density ;

e) equally important, estimate the error due to finite spectral length.

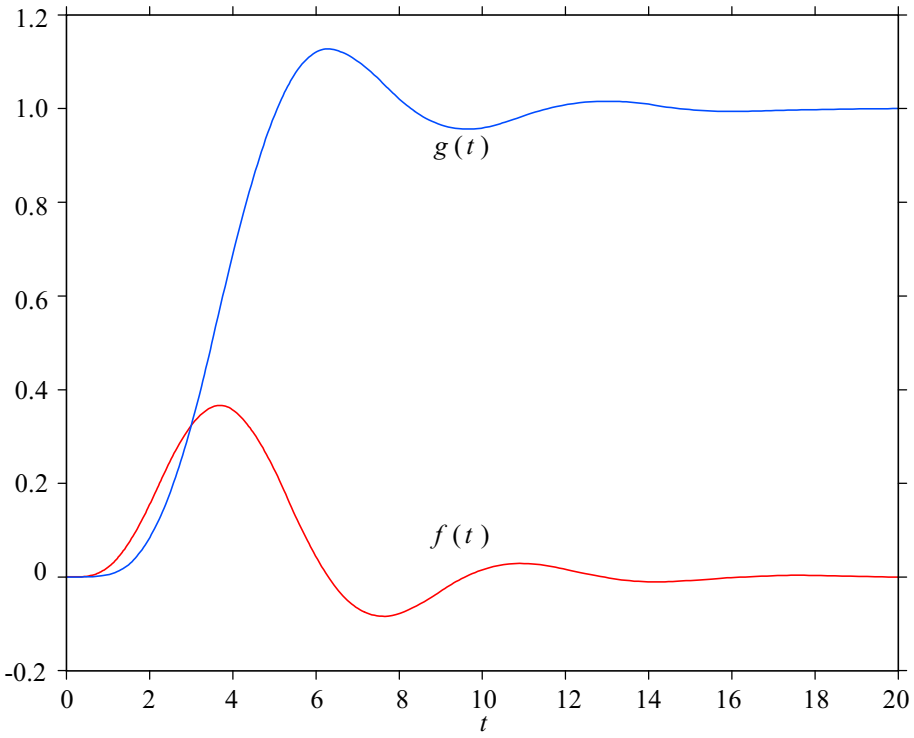


Fig. 6.5.1 : The impulse- and step-response of the 5th-order Butterworth system. The impulse amplitude has been normalized to represent the response to an ideal, infinitely narrow, infinite amplitude input impulse. The impulse-response reaches the peak value at the time equal to the envelope-delay value at d.c.; this delay is also the half-amplitude delay of the step-response. The step-response first crosses the final value at the time equal to the envelope-delay maximum. Also the step-response peak value is reached when the impulse-response crosses the zero-level for the first time. If the impulse-response is normalized to have the area (the sum of all samples) equal to the system d.c. gain, the step-response would be simply a time-integral of it.

6.5.1. Impulse Response, Using FFT

The basic idea behind this method is the fact that the Fourier transform is a special case of the more general Laplace transform and the Dirac impulse function is such a special type of signal for which the Fourier-transform solution always exists. Comparing Eq. 1.3.8 and Eq. 1.4.3 and taking in account that $s = \sigma + j\,\omega$, we see :

$$\mathcal{L}\{f(t)\} = \mathcal{F}\{f(t)\,\mathrm{e}^{-\sigma t}\} \tag{6.5.1}$$

Since the complex-plane variable s is composed of two independent parts (real and imaginary), then $F(s)$ may be treated as a function of two independent variables. This can be most easily understood by looking at Fig. 6.4.1, where the complex-frequency response (magnitude) of a 5-pole Butterworth function is plotted as a 3-D surface over the Laplace plane.

In that particular case we had :

$$F(s) = \frac{-\,s_1 s_2 s_3 s_4 s_5}{(s-s_1)\,(s-s_2)\,(s-s_3)\,(s-s_4)\,(s-s_5)} \tag{6.5.2}$$

where $s_{1\text{-}5}$ have the same values as in the example at the beginning of Sec. 6.4.1 .

When s in Eq. 6.5.2 gets closer to the value of one of the poles, s_i , then $|F(s)|$ increases until becoming infinitely large for $s = s_i$.

Let us now introduce a new variable p such that :

$$p = s\,\Big|_{\sigma\,=\,0} \qquad \text{or:} \qquad p = j\omega \tag{6.5.3}$$

This has the effect of cutting the surface along the imaginary axis, as we did in Fig. 6.4.1, exposing the curve on the surface along the cut, which is $|F(j\omega)|$, or in words : the complex-frequency-response magnitude, $M(\omega)$. As we have indicated in Fig. 6.4.5, we show it usually in log-log scaled plot. For the purpose of transient response calculus, however, a linear frequency scale is appropriate (as in Fig. 6.4.2), since we need the result of the inverse transform in linear time-scale increments.

Now as we have established the connection between the Laplace-transformed transfer function and its frequency response, we have another point to consider : conventionally, the Fourier transform is used to calculate waveform spectra, so we need to establish the relationship between a frequency response and a spectrum. Also, we must explore the effect of taking **discrete values (sampling)** of the time-domain and frequency-domain functions, and see to what extent we **approximate** our **results** by taking **finite-length vectors of finite-density sampled data.** Those readers, who would like to embed the inverse transform in a microprocessor controlled instrument, will have to pay attention to **amplitude quantization (finite word length)** as well, but in Matlab this is not an issue.

We have examined the Dirac function $\delta(t)$ and its spectrum in Part 1, Sec. 1.6.6. Note that the spectral components are separated by $\Delta\omega = 2\,\pi/T$, where T is the impulse repetition period. If we allow $T \to \infty$ then $\Delta\omega \to 0$. In these conditions we can hardly speak of discrete spectral components as the spectrum has

become very dense; we rather speak of **spectral density.** Also, instead of individual component's magnitude, we speak of **spectral envelope** and for $\delta(t)$ it is essentially flat.

However, if we do not have an infinitely dense spectrum, then $\Delta\omega$ is small, but not 0, and this merely means that the impulse repeats after a finite period $T = 2\pi/\Delta\omega$ (this is the mathematical equivalent of testing a system by an impulse of a duration much shorter than the smallest system time-constant and of a repetition period much larger than the largest system time-constant).

Now let us take such an impulse and present it to a system having a selective frequency response. Fig. 6.5.2 shows the results both in time- and frequency-domain (magnitude). The time-domain response is obviously the system impulse-response, and its equivalent in the frequency-domain is a spectrum, whose density is equal to the input spectral density, but with the spectral envelope shaped by the system frequency response. The conclusion is that we only have to sample the frequency response at some finite number of frequencies and perform a discrete-Fourier-transform inversion to obtain the impulse response.

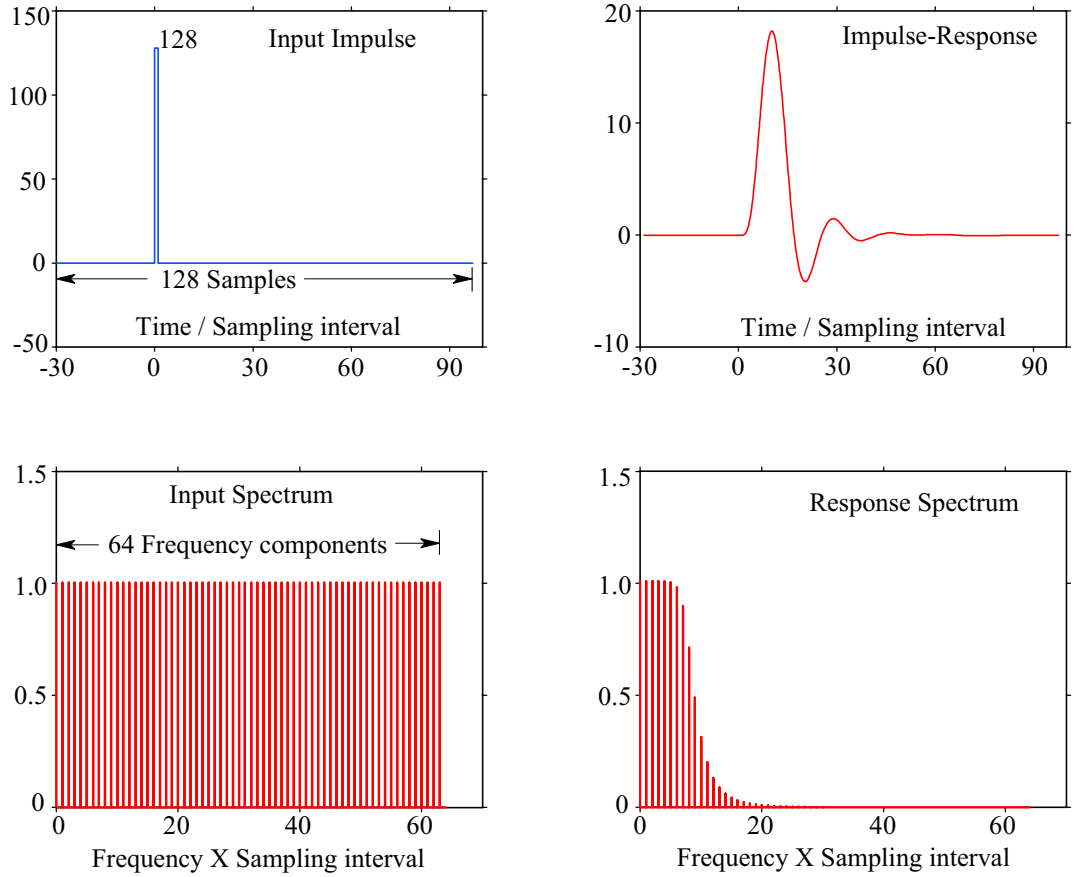


Fig. 6.5.2 : Time-domain and frequency-domain representation of a 5-pole Butterworth system impulse response. The spectral envelope (only the magnitude is shown here) of the output is shaped by the system frequency response, while the spectral density remains unchanged. From this fact we conclude that the time-domain response can be found from a system frequency response using inverse Fourier transform. Horizontal scale is the number of samples (128 in the time-domain and 64 in the frequency-domain - see text for explanation).

If we know the magnitude and phase response of a system at some finite number of equally spaced frequency points, then each point represents :

$$F_i = M_i \, cos\,(\omega_i \, t - \varphi_i) \tag{6.5.4}$$

As the contribution of frequencies which are attenuated by more than, say, 60 dB can be neglected, we don't have to take into account an infinitely large number of frequencies, and the fact that we don't have an infinitely dense spectrum merely means that the input impulse repeats in time. Then, applying the superposition theorem, the output is equal to the sum of all the separate frequency components.

Thus for each time point the computer must perform the addition :

$$f\,(t_k) = \sum_{i\,(\omega_{min})}^{i\,(\omega_{max})} M_i \, cos\,(\omega_i \, t_k - \varphi_i) \tag{6.5.5}$$

Eq. 6.5.5 is the **discrete Fourier transform,** with the exponential part expressed in trigonometric form. However, if we would plot the response calculated after Eq. 6.5.5, we could see that the time-axis is reversed and, from the theory on Fourier-transform properties (symmetry property, [Ref. 6.7, p.192]), we know that the application of two successive Fourier transforms returns the original function but with reversed sign of the independent variable :

$$\mathcal{F}\Big\{\mathcal{F}\{f(t)\}\Big\} = \mathcal{F}\{F(j\omega)\} = f(-t) \tag{6.5.6}$$

or more generally :

$$f(t) \underset{\mathcal{F}^{-1}}{\overset{\mathcal{F}}{\rightleftarrows}} F(j\omega) \underset{\mathcal{F}^{-1}}{\overset{\mathcal{F}}{\rightleftarrows}} f(-t) \underset{\mathcal{F}^{-1}}{\overset{\mathcal{F}}{\rightleftarrows}} F(-j\omega) \underset{\mathcal{F}^{-1}}{\overset{\mathcal{F}}{\rightleftarrows}} f(t) \tag{6.5.7}$$

The main drawback in using Eq. 6.5.5 is the high total number of operations, as there are three input data vectors of equal length (ω, M, φ) and each contributes to every time-point result. It seems that greater efficiency might be obtained by using the input frequency-response data in the complex form, with the frequency vector represented by the index of the $F(j\omega)$ vector.

Now $F(j\omega)$ in its complex form is a two-sided spectrum, as it was shown in Fig. 6.4.3, and we are often faced with only a single-sided spectrum. It can be shown that a real-valued $f(t)$ will always have $F(j\omega)$ symmetrical about the real axis σ. Thus :

$$F_N(j\omega) = F_P^*(-j\omega) \tag{6.5.8}$$

F_N and F_P are the $\omega < 0$ and $\omega > 0$ parts of $F(j\omega)$, with their inverse transforms labeled $f_N(t)$ and $f_P(t)$.

This symmetry property follows from the definition of the **negative-frequency concept** : instead of having a single phasor rotating counter-clockwise (positive by definition) in the complex plane, we can always have two half-amplitude phasors rotating in opposite directions at the same frequency (as we have already seen it drawn in Part 1 , Fig. 1.1.1 ; for vector analysis, see Fig. 6.5.3). We can therefore conclude that

the inherent conjugate symmetry of the complex plane allows us to define "negative frequency" as a clokwise-rotating, half-amplitude phasor, being the complex-conjugate of the usual counter-clockwise (positive by definition) rotating (but now also half-amplitude) phasor. And this is not just a fancy way of making simple things complex, but is rather a direct consequence of our dislike of sine-cosine representation and the preference for the complex-exponential form, which is much simpler to handle analytically.

One interesting aspect of the negative-frequency concept is the ***Shannon* sampling theorem**: for a continuous signal, sampled with a frequency f_s, no additional information can be extracted from the frequency range between $f_s/2$ and f_s, since the resulting spectrum is symmetrical about $f_s/2$. This means that a frequency equal to f_s can not be distinguished from a d.c. level and any frequency in between, $f_s/2 < f < f_s$, can not be distinguished from $f_s - f$. The $f_s/2$ is called the *Nyquist frequency*.

But please, also note that this "negative frequency" does not necessarily imply "negative time", since the negative time is defined as the time before some arbitrary instant $t = 0$ at which the signal was applied. In contrast, the negative-frequency response is just one-half of the full description of the $t \geq 0$ signal.

However, those readers who will explore the properties of the *Hilbert-transform* will learn that this same concept can be extended to the $t < 0$ signal region, but this is beyond the scope of this text.

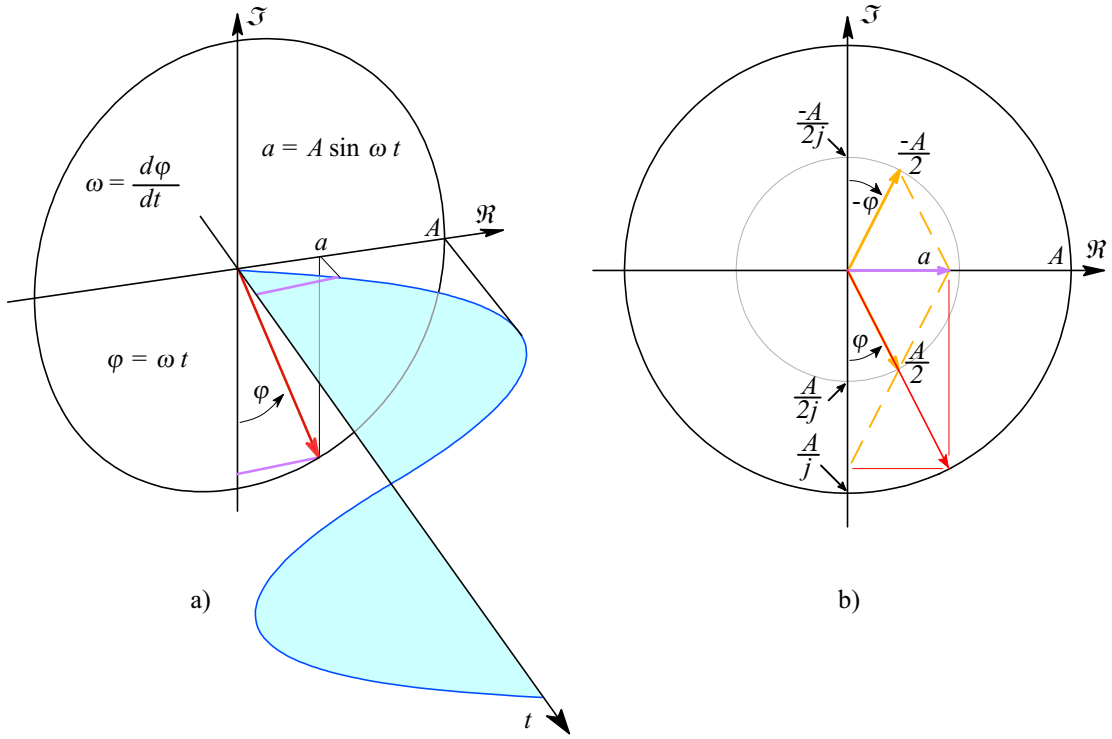


Fig. 6.5.3 : As in Part 1, Fig. 1.1.1, but in a slightly different perspective: **a)** the real signal instantaneous amplitude $a(t) = A \sin\varphi$, where $\varphi = \omega t$. **b)** the real part of the instantaneous signal phasor, $\Re\{\vec{0A}\} = \vec{0a} = A \sin\varphi$, can be decomposed into two half-amplitude, opposite rotating, complex-conjugate phasors, $-\frac{A}{2j}\sin\varphi + \frac{A}{2j}\sin(-\varphi)$. The second term has rotated by $-\varphi = -\omega t$ and, since t is obviously positive (see the **a)** graph), the negative sign is attributed to ω; thus, **clockwise rotation is interpreted as a "negative frequency"**!

Eq. 6.5.8 can thus be used to give :

$$F(j\omega) = F_P(j\omega) + F_P^*(-j\omega) \tag{6.5.9}$$

but from Eq. 6.5.7 :

$$\mathcal{F}^{-1}\Big\{ F_P^*(-j\omega)\Big\} = f_P^*(t) \tag{6.5.10}$$

hence, using Eq. 6.5.9 and Eq. 6.5.10 and taking into account the cancellation of imaginary parts, we get :

$$f(t) = f_P(t) + f_P^*(t) = 2\,\Re\Big\{ f_P(t)\Big\} \tag{6.5.11}$$

Eq. 6.5.11 means that if $F_P(j\omega)$ is the $\omega \geq 0$ part of the Fourier-transformed real-valued function $f(t)$, its Fourier-transform inversion $f_P(t)$ will be a complex function whose real part is equal to $f(t)/2$. Summing the complex-conjugate pair results in a doubled real-valued $f(t)$. So, by Eq. 6.5.10 and Eq. 6.5.11, we can calculate the system impulse response from just one-half of its complex-frequency response **using forward Fourier transform** (not inverse!) :

$$f(t) = 2\,\Re\left\{ \left[\mathcal{F}\Big\{ F_P^*(j\omega)\Big\} \right]^* \right\} \tag{6.5.12}$$

Note that the second (outer) complex-conjugate is here only to satisfy the mathematical consistency - in the actual algorithm it can be safely omitted, since only the real part is required.

As the operator $\mathcal{F}\{\,\}$ in Eq. 6.5.12 implies integration, we must use the **discrete-Fourier-transform (DFT)** for computation. The DFT can be defined by decomposing the Fourier-transform integral into a finite sum of N elements :

$$F(k) = \frac{1}{N} \sum_{i=0}^{N-1} f(i)\, \mathrm{e}^{-j\frac{2\pi k}{N}} \tag{6.5.13}$$

That means going again through a large number of operations, comparable to Eq. 6.5.5. Instead, we can apply the **fast-Fourier-transform (FFT)** algorithm and, owing to its excellent efficiency, save the computer a great deal of work.

Cooley and *Tukey* [Ref. 6.8] have shown that if $N = 2^B$ and B integer, there is a smart way to use Eq. 6.5.13, due to the periodical nature of the Fourier transform. If Eq. 6.5.13 is expressed in a matrix form, then the matrix, which represents its exponential part, can be divided into its even and odd part and the even part can be assigned to $N/2$ elements. The remaining part can then also be divided as before and the same process can then be repeated over and over, so that we end-up with a number ($\log_2 N$) of individual sub-matrices. Further, it can be shown that these sub-matrices contain only two non-zero elements, one of them being always unity (1 or j). So, multiplying by each of the factor matrices requires only N complex multiplications.

Finally (or firstly, depending on whether we are using the decimation-in-frequency or decimation-in-time algorithm), we rearrange the data, by writing the position of each matrix element in a binary form and then reversing the binary digits ("reshuffling"). The total number of multiplications is thus $N \log_2 N$ instead of the N^2 required to multiply in one step. Other operations are simple and fast to execute (addition, change of sign, change of order). Thus, in the case of $B = 10$, $N = 1024$, and $N \log_2 N = 10240$, while $N^2 = 1048576$, **so a two orders of magnitude reduction** in number of multiplications is achieved.

Matlab has a command named "FFT" which uses the radix-2 type algorithm and we will use it as it is. Those readers, who would like to implement the FFT algorithm by themselves, can find the detailed treatment in [Ref. 6.8, 6.9 and 6.10].

A property of the FFT algorithm is that it returns the spectrum of a real-valued signal as folded about the Nyquist frequency (one-half of the frequency at which the signal was sampled). As we have seen in Fig. 6.5.2, if we have taken 128 signal samples, the FFT returns the first 64 spectral components from $\omega = 0, 1, 2, \cdots, 63$ but then the remaining 64 components, which are the complex-conjugate of the first ones, in the reversed order. This is in contrast to what we were used to, as we expect the complex-conjugate part to be on the $\omega < 0$ side of the spectrum. On the other hand, this is equivalent, since the 128 points taken in the signal time-domain window are implicitly assumed to repeat and, consequently, the spectrum is also repeating every 128 points. So, if we use the standard inverse-FFT procedure, we must take into account all 128 spectral components to get only 64 points of the signal back. However, note that the Eq. 6.5.12 **requires only a single-sided spectrum of N points** to return N points of the impulse response. This is, clearly, an **additional two-fold improvement** in algorithm efficiency.

To continue with our 5th-order Butterworth example, we can now calculate the impulse and step response by using the TRESP routine :

```
[z,p]=buttap(5);          % the 5th-order Butterworth system poles
w=(0:1:255)/8;            % form a linearly spaced frequency vector
F=freqw(z,p,w);           % the frequency response at w
[I,t]=tresp(F,w,'i');     % I : ideal impulse, t : normalized time
S=tresp(F,w,'s');         % S : step response ( time same as for I )
plot(t(1:100),I(1:100),t(1:100),S(1:100))
                          % plot 100 points of I and S vs. t
```

The results should look just like Fig. 6.5.1 . Here is the TRESP routine :

```
function [y,t]=tresp(F,w,r,g)
%TRESP   Transient RESPonse, using Fast-Fourier-Transform algorithm.
%    Call : [y,t]=tresp(F,w,r,g);
%    where:
%     F --> complex-frequency response, length-N vector, N=2^B,
B=int.
%     w --> can be the related frequency vector of F, or it
%           can be the normalized frequency unit index, or it
%           can be zero and the n.f.u. index is found from F.
%     r --> a character, selects the response type returned in y:
%           - 'u' is the unity-area impulse response (the default)
%           - 'i' is the ideal impulse response
%           - 's' is the step response
%     g --> an optional input argument: plot the response graph.
%     y --> the selected system response.
%     t --> the normalized time-scale vector.

% Author :  Erik Margan, 880414, Last rev. 000310, Free of copyright!

% ----------- Preparation and checking the input data ------------
if nargin < 3
      r='u';              % select the default response if not
specified
end
G=abs(F(1));              % find system d.c. gain
N=length(F);              % find number of input frequency samples
v=length(w);              % get the length of w
if v == 1
      m=w;                % w is the normalized frequency unit or zero
elseif v == N
      m=find(abs(w-1)==min(abs(w-1)))-1;  % find the norm. freq. unit
      if isempty(m)
            m=0;          % not found, try from the half-power
bandwidth
      end
else
      error('F and w are expected to be of equal length !');
end
if m == 0
      m=max(find(abs(F)>=G/sqrt(2)))-1;   % find the n.f.u. index
end
% check magnitude slope between the 2nd and 3rd octave above cutoff
M=abs(diff(20*log10(abs(F(1+4*m*[1,2])))));
x=3;                      % system is 3rd-order or higher (>=18dB/2f)
if M < 15
      x=2;                % probably a 2nd-order system (12dB/2f)
elseif M < 9
      x=1;                % probably a 1st-order system (6dB/2f)
end
```

```
% ----------- Form the window function --------------------------
if x < 3
      W=0.54-0.46*cos(2*pi*(N+1:2*N)/(2*N)); % right-half Hamming
      F=W.*F;           % frequency response windowed
end

% ----------- Normalize the time-scale --------------------------
A=2*pi*m;               % amplitude denormalization factor
dt=A/N;                 % calculate delta_t
if v == N
      dt=dt/w(m+1);     % correct for the actual frequency unit
end
t=dt*(0:1:N-1);         % form the normalized time scale

% ----------- Calculate the impulse response --------------------
y=2*real(fft(conj(F)))-G;     % calculate iFFT and null d.c. offset
if x == 1
      er1=A*G-y(1);     % fix the 1st-point error for 1st-order
system
      y(1)=A*G;
end
if r == 'u' | r == 'U' | r == 's' | r == 'S'
      y=y/N;            % normalize area to G
end

% ----------- Calculate the step response -----------------------
if r == 's' | r == 'S'
      if x == 1
            y=y*(1/(1+er1/N)); % correct 1st-point error
      end
      y=cumsum([0, y]); % integrate to get the step-response
      if x > 1
            y=(y(1:N)+y(2:N+1))/2;  % compensate half-sample t-delay
            y(1)=0;
      else
            y=y(1:N);
      end
end

% ----------- Normalize the amplitude to ideal ------------------
if r == 'i' | r == 'I'
      y=y/A;            % denormalize impulse amplitude
end

% ----------- Plot the graph ------------------------------------
if nargin == 4
      plot( t, y, '-r' ), xlabel('Time')
      if r == 'i' | r == 'I' | r == 'u' | r == 'U'
            title('Impulse-response')
      else
            title('Step-response')
      end
end
```