RUTGERS UNIVERSITY
School of Engineering – ECE Department
332:447 – Digital Signal Processing Design
S. J. Orfanidis – Fall 2017

Project 2 – Dynamic Range Control
Compressors, Limiters, Expanders, and Noise Gates

Assigned 09/21/17 – Due 09/28/17

1. Introduction

The purpose of this project is to design and implement a dynamic range control system that can be used as a compressor, limiter, expander, or noise gate. Moreover, it can be used as "ducker".

It is based on Section 8.2.5 of the I2SP text [1]. Please review the examples in that section before you start on this project. Additional information on dynamics processors may be found in the references [2–7].

Compressors, limiters, expanders, and gates have a wide variety of uses in audio signal processing. Compressors attenuate strong signals; expanders attenuate weak signals. Because they affect the dynamic range of signals, they are referred to as *dynamics processors*.

Compressors are used mainly to *decrease* the dynamic range of audio signals so that they fit into the dynamic range of the playback or broadcast system; for example, for putting a recording on audio tape. But there are several other applications, such as announcers "ducking" background music, "de-essing" for eliminating excessive microphone sibilance, and other special effects [2].

Expanders are used for *increasing* the dynamic range of signals, for noise reduction, and for various special effects, such as reducing the sustain time of instruments [2].

A typical steady-state input/output relationship for a compressor or expander is as follows, in absolute and decibel units:

$$y = y_0 \left(\frac{x}{x_0}\right)^{\rho} \quad \Rightarrow \quad 20\log_{10}\left(\frac{y}{y_0}\right) = \rho \cdot 20\log_{10}\left(\frac{x}{x_0}\right) \tag{1}$$

where x is here a constant input, x_0 a desired threshold, and ρ defines the compression or expansion ratio.[†] A compressor is effective only for $x \geq x_0$ and has $\rho < 1$, whereas an expander is effective for $x \leq x_0$ and has $\rho > 1$. Fig. 1 shows these relationships in dB, so that a 1 dB change in the input causes ρ dB change in the output, that is, ρ is the slope of the input/output straight lines. The hard-knee change in slope can be replaced by a soft-knee by quadratically interpolating between the two slope lines [5], but we will not implement this here.

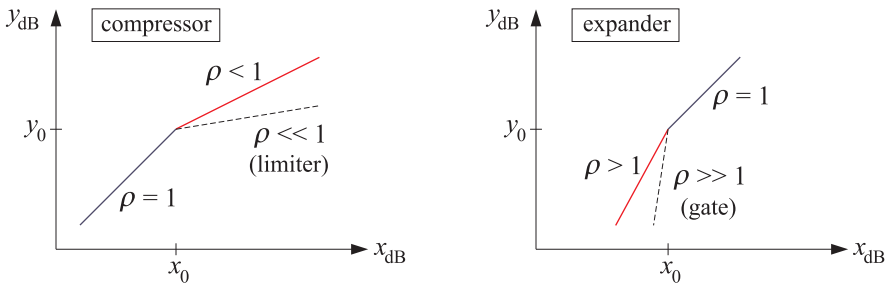

Fig. 1 Static input/output relationship of compressor or expander.

[†]The inverse quantity, $R = 1/\rho$, is commonly called the "compression ratio," whereas ρ is the "slope".

Typical practical values are $\rho = 1/4$–$1/2$ for compression, and $\rho = 2$–4 for expansion. *Limiters* are extreme forms of compressors that prevent signals from exceeding certain maximum thresholds; they have very small slope $\rho \ll 1$, for example, $\rho = 1/10$. *Noise gates* are extreme cases of expanders that infinitely attenuate weak signals, and therefore, can be used to remove weak background noise; they have very large slopes $\rho \gg 1$, for example, $\rho = 10$.

The I/O equation (1) is appropriate only for constant signals. Writing $y = Gx$, we see that the effective gain of the compressor is a nonlinear function of the input of the form $G = G_0 x^{\rho-1}$. For time-varying signals, the gain must be computed from a *local average* of the signal which is representative of the signal's level.

A model of a compressor/expander is shown in Fig. 2. The level detector generates a *control signal* c_n that controls the gain g_n of the multiplier through a nonlinear gain processor.

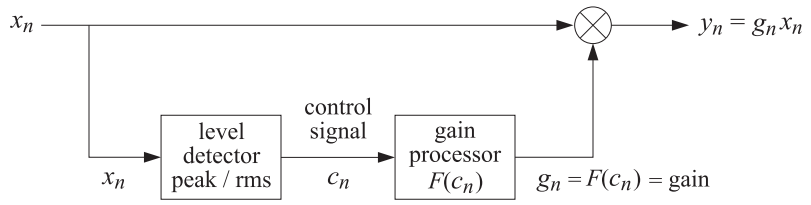

Fig. 2 Compressor/expander dynamics processor.

Depending on the type of compressor, the control signal may be (1) the instantaneous *peak* value $|x_n|$, (2) the *envelope* of x_n, or (3) the *root-mean-square* value of x_n. A simple model of the envelope detector is as follows, with $0 < \lambda < 1$,

$$c_n = \lambda\, c_{n-1} + (1-\lambda)\,|x_n| \qquad \text{(level detector)} \tag{2}$$

The difference equation for c_n acts as a *rectifier* followed by a simple first-order lowpass EMA[†] filter. The transfer function, impulse and unit-step responses of this filter are,

$$H(z) = \frac{1-\lambda}{1-\lambda z^{-1}}, \qquad h_n = (1-\lambda)\,\lambda^n u(n), \qquad (h * u)_n = \left(1-\lambda^{n+1}\right) u(n) \tag{3}$$

Thus the filter output responds exponentially quickly to a level change, with an effective time constant, $t_{\text{eff}} = n_{\text{eff}} T_s$, where n_{eff} is the number of samples to converge to within ϵ of the final level, that is, $1 - \lambda^{n_{\text{eff}}} = 1 - \epsilon$, or,

$$\lambda^{n_{\text{eff}}} = \epsilon \quad \Rightarrow \quad n_{\text{eff}} = \frac{\ln \epsilon}{\ln \lambda} \quad \Rightarrow \quad \lambda = \epsilon^{1/n_{\text{eff}}} = \epsilon^{T_s/t_{\text{eff}}} \tag{4}$$

and ϵ is a user-definable parameter, such as, $\epsilon = 0.1,\ 0.01,\ 0.001,\ 0.05$, corresponding respectively to the so-called 20-dB, 40-dB, 60-dB, or 95% time constants. For these particular values of ϵ, Eq. (4) for calculating the filter parameter λ can be written in the approximate equivalent exponential forms,

$$\begin{aligned} \epsilon &= [0.1,\ 0.01,\ 0.001,\ 0.05] \approx \left[e^{-2.3},\ e^{-4.6},\ e^{-6.9},\ e^{-3}\right] \\ \lambda &= \left[e^{-2.3T_s/t_{\text{eff}}},\ e^{-4.6T_s/t_{\text{eff}}},\ e^{-6.9T_s/t_{\text{eff}}},\ e^{-3T_s/t_{\text{eff}}}\right] \end{aligned} \tag{5}$$

The time constant, t_{eff}, controls the time to rise or fall to a new input level. The time to rise to a level above the threshold (where the compressor is active) is called the *attack* time constant. The time to drop to a level below the threshold (where the compressor is inactive) is called the *release* time. In audio applications, t_{eff} is typically specified in milliseconds.

For $\lambda = 0$, Eq. (2) becomes an instantaneous peak detector. This case is useful when the compressor is used as a limiter. If in Eq. (2) the absolute value $|x_n|$ is replaced by its square, $|x_n|^2$, the control signal will track the *mean-square* value of the input. In this case, the quantity $\sqrt{c_n}$ will track the RMS level of the input.

[†] exponentially-weighted moving average

The gain processor is a nonlinear function of the control signal imitating the I/O equation (1). For a compressor, we may define the gain function to be:

$$g = F(c) = \begin{cases} \left(\dfrac{c}{c_0}\right)^{\rho-1}, & \text{if } c \geq c_0 \\ 1, & \text{if } c \leq c_0 \end{cases} \qquad \text{(compressor, limiter)} \tag{6}$$

where c_0 is a desired *threshold* and $\rho < 1$. For an expander, we have $\rho > 1$ and:

$$g = F(c) = \begin{cases} 1, & \text{if } c \geq c_0 \\ \left(\dfrac{c}{c_0}\right)^{\rho-1}, & \text{if } c \leq c_0 \end{cases} \qquad \text{(expander, gate)} \tag{7}$$

Thus, the gain g_n and the final output signal y_n are computed as follows:

$$\boxed{\begin{aligned} g_n &= F(c_n) \\ y_n &= g_n x_n \end{aligned}} \tag{8}$$

Compressors/expanders are examples of *adaptive signal processing* systems, in which the filter coefficients (in this case, the gain g_n) are time-dependent and adapt themselves to the nature of the input signals. The level detector (2) serves as the "adaptation" equation and its attack and release time constants are the "learning" time constants of the adaptive system; the parameter λ is called the "forgetting factor" of the system.†

In this project, you will implement a more realistic compressor system depicted in Fig. 3 that incorporates the following features: (1) It can accommodate different attack and release time constants, quantified by different filter λ-parameters, say, λ_a, λ_r. (2) The nonlinear gain $g_n = F(c_n)$ is smoothed further by a lowpass filter, which can be taken to be either an FIR averager or another EMA filter like the detector. (3) An overall delay may be introduced at the output that helps reduce transient overshoots. (4) Optionally, the gain-control signal c_n could be calculated not from the given input x_n, but from an alternative input as would be the case in "ducking" applications, or by a bandpass-filtered version of x_n, as in "de-essing" applications.

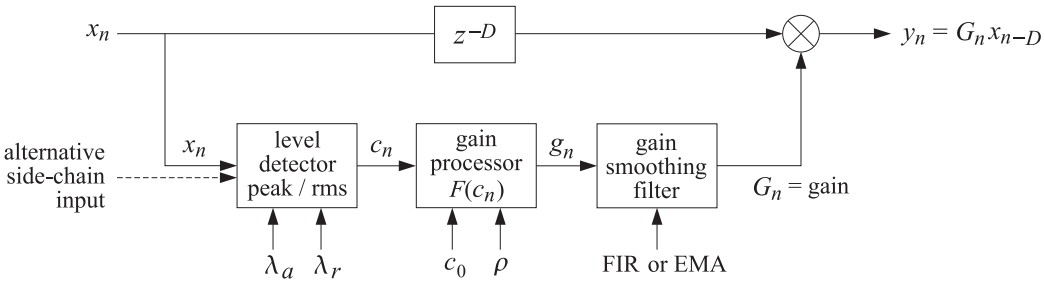

Fig. 3 Compressor/expander dynamics processor.

The sample-by-sample processing algorithm may be summarized as follows:

$$\boxed{\begin{aligned} &\text{for each audio sample } x_n, \text{ do,} \\ &\quad 1.\ \text{calculate } c_n, \text{ using } \lambda_a \text{ and } \lambda_r \\ &\quad 2.\ \text{calculate nonlinear gain, } g_n = F(c_n) \\ &\quad 3.\ \text{calculate smoothed gain, } G_n \\ &\quad 4.\ \text{calculate output, } y_n = G_n x_{n-D} \end{aligned}} \tag{9}$$

These steps are outlined below. To accommodate different attack and release time constants, Eq. (2) may be replaced by the following version which switches from the attack mode λ_a to the

†We will be discussing such adaptive systems under the Signal Extraction and Adaptive Filtering course topics.

release mode λ_r depending on the calculated level,

$$c_n = \begin{cases} \lambda_a c_{n-1} + (1-\lambda_a)\,|x_n|\,, & \text{if} \quad |x_n| \geq c_{n-1} \\ \lambda_r c_{n-1} + (1-\lambda_r)\,|x_n|\,, & \text{if} \quad |x_n| < c_{n-1} \end{cases} \tag{10}$$

By choosing $\lambda_a < \lambda_r$, the attack time constant will be shorter than the release one, allowing a quicker response at the onset of compression or expansion. Next, the gain g_n is computed from Eqs. (6) or (7), and then passed into the smoothing filter, which may taken to be either an FIR averager or an EMA filter of the form,

$$\begin{aligned} G_n &= \frac{1}{L}\left[g_n + g_{n-1} + \cdots + g_{n-L+1}\right] \qquad \text{(FIR)} \\ G_n &= \lambda\, G_{n-1} + (1-\lambda)\, g_n \qquad\qquad\quad \text{(EMA)} \end{aligned} \tag{11}$$

The two filters behave approximately equivalently if the parameters L, λ are chosen such that:[†]

$$L \approx \frac{1+\lambda}{1-\lambda} \quad \Leftrightarrow \quad \lambda \approx \frac{L-1}{L+1} \tag{12}$$

In some implementations [3,4], the EMA smoother is chosen to have different attack and release time constants as follows—however, the simpler versions of Eq. (11) will be used in this project,

$$G_n = \begin{cases} \lambda_a G_{n-1} + (1-\lambda_a)\, g_n\,, & \text{if} \quad g_n \geq g_{n-1} \\ \lambda_r G_{n-1} + (1-\lambda_r)\, g_n\,, & \text{if} \quad g_n < g_{n-1} \end{cases} \tag{13}$$

2. Design Procedures

a. Consider a sinusoid $x(t) = A\cos(2\pi f t)$. Show that its mean-square average over one period, and its absolute average are given by:

$$\overline{x^2(t)} = \frac{1}{2}A^2\,, \quad \overline{|x(t)|} = \frac{2}{\pi}A \tag{14}$$

These could be used as guides in choosing the compressor thresholds.

b. It is desired to design a digital dynamics processor based on the block diagram of Fig. 3 that operates at an 8 kHz sampling rate and has 20-dB attack and release time constants of 2 msec and 10 msec, respectively.

Calculate the corresponding values of the forgetting factors λ_a, λ_r, and use them in Parts (c–g) of this project. In implementing the gain smoothing filter, choose its FIR length L or its EMA parameter λ based on the attack value of λ_a.

c. Generate a 75-msec long input signal $x(t)$ sampled at 8 kHz, consisting of three sinusoids of the following frequencies, amplitudes, and durations,

$$\begin{aligned} f_1 &= 0.3 \text{ kHz}\,, \quad A_1 = 2.0\,, \quad \text{duration}, \quad 0 \leq t < 25 \text{ msec} \\ f_2 &= 0.6 \text{ kHz}\,, \quad A_2 = 4.0\,, \quad \text{duration}, \quad 25 \leq t < 50 \text{ msec} \\ f_3 &= 1.2 \text{ kHz}\,, \quad A_3 = 0.5\,, \quad \text{duration}, \quad 50 \leq t < 75 \text{ msec} \end{aligned}$$

Calculate the mean-absolute values of the three levels. Plot $x(t)$ versus t.

[†] to be explained later under the Signal Extraction course topics

d. Let x_n denote the time samples of $x(t)$ and use them as the input to Fig. 3. Using the parameters, $\rho = 1/3,\ c_0 = 1,\ D = 0$, calculate the corresponding compressed signal y_n using an FIR gain smoothing filter. Plot $y(t)$ vs. sampled t using the same scales as for $x(t)$.

   Moreover, in three separate graphs, plot versus t the control signal $c(t)$, the raw gain $g(t)$, and its smoothed version $G(t)$.

   Notes: Do not use the built-in function **filter** in this part, rather, implement the algorithm of Eq. (9) on a sample by sample basis.

   Although Eqs. (6) and (7) can be combined into a one-line anonymous vectorized MATLAB function, such function would generate NaN's if the control signal happened to be zero, $c = 0$. Therefore, it is best to write a separate function which handles such circumstance.

e. For the same input x_n, choose appropriate values for ρ, c_0 such that the compressor would act as a limiter that limits the f_2 signal, but not f_1 and f_3. You may use an FIR or EMA gain smoother in this part.

   Plot the corresponding output $y(t)$ using the same scales as for $x(t)$, and on separate graphs also plot the signals $c(t), g(t), G(t)$.

   For easy reference place on the $y(t)$ graph the values of ρ, c_0 that you used.

f. Next, choose appropriate values for ρ, c_0 and an FIR or EMA smoother, such that the dynamics processor would act as an expander that attenuates the f_3 component only, while leaving f_1, f_2 unaffected.

   Plot the corresponding output $y(t)$ using the same scales as for $x(t)$, and on separate graphs also plot the signals $c(t), g(t), G(t)$.

   For easy reference place on the $y(t)$ graph the values of ρ, c_0 that you used.

g. Repeat part (f) such that the dynamics processor would act as a noise gate suppressing the f_1 and f_3 components, but not f_2.

   Then, repeat this part, so that now the noise gate removes only f_3, but not f_1, f_2.

h. In this part, you will implement a dynamics processor acting as a "ducker", shown in Fig. 4. Please load the following wave files into MATLAB,

```
[xs,fs] = audioread('speech.wav');    % speech signal
[xm,fs] = audioread('music.wav');     % music signal
```

   The signals $x_s(t)$ and $x_m(t)$ both have exactly the same duration of approximately 7 seconds, and the same sampling rate of $f_s = 44.1$ kHz. The actual speech contained in $x_s(t)$ is preceded and anteceded by 2 seconds of silence. Plot the signals $x_s(t)$ and $x_m(t)$ versus t, as well as the combined mixed signal, $x(t) = x_s(t) + x_m(t)$ (not shown in Fig. 4).

   In the dynamics processor of Fig. 4, the gain $G(t)$ to be applied to the music signal $x_m(t)$ is controlled by the side-chain input speech signal $x_s(t)$. The scaled music signal output

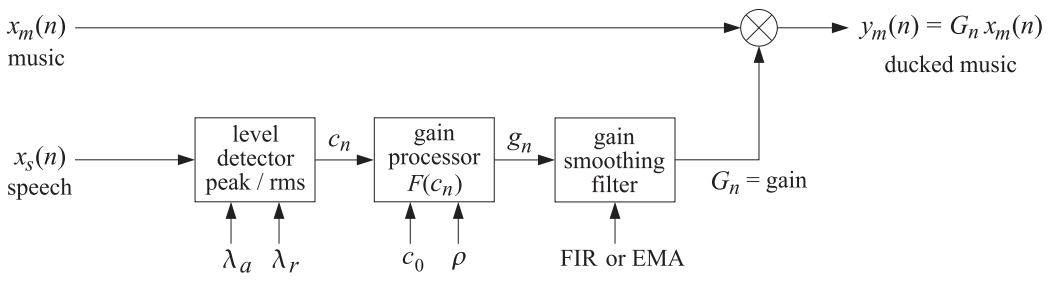

Fig. 4 Compressor/ducker dynamics processor.

is $y_m(t) = G(t)x_m(t)$. The compressor parameters are chosen such that $G(t)$, and hence $y_m(t)$, become very small whenever the speech signal $x_s(t)$ is present, thus, the music signal is "ducked" allowing a clearer hearing of the speech in the combined mixed output,

$$y(t) = x_s(t) + y_m(t) = x_s(t) + G(t)x_m(t) \tag{15}$$

Calculate the combined speech plus ducked music signal, $y(t) = x_s(t) + y_m(t)$, listen to it, and compare it with the unprocessed combined signal, $x(t) = x_s(t) + x_m(t)$.

Experiment with different values of the parameters $\rho, c_0, \lambda_a, \lambda_r$ until you are satisfied with the ducking result (you may use an EMA gain smoother here). As a starting point, you may want to choose c_0 to be somewhere between 40–60 dB below the maximum value of $x_s(t)$ and choose the attack time constant to be a few tens of milliseconds, and the release time constant, a few hundreds of milliseconds.

Once you are satisfied with the results, plot $y(t)$ and $y_m(t)$ versus t, as well as the control and gain signals, $c(t), g(t), G(t)$.

Save your processed speech plus ducked music signal $y(t)$ in a wave file and upload it to Sakai with your report.

For your reference, I am including the following wave files containing $x(t) = x_s(t) + x_m(t)$, and my versions of the signals $y_m(t)$ and $y(t) = x_s(t) + y_m(t)$, which are also depicted at the end.

$x(t) = x_s(t) + x_m(t)$,	`'speech+music.wav'`
$y_m(t) = G(t)x_m(t)$,	`'ducked music.wav'`
$y(t) = x_s(t) + y_m(t)$,	`'speech+ducked.wav'`

3. References

All references below may be found on Sakai Resources.

[1] S. J. Orfanidis, *Introduction to Signal Processing*, online book, 2010.

[2] B. Hurtig, "The Engineer's Notebook: Twelve Ways to Use Dynamics Processors," *Electronic Musician*, **7**, no. 3, 66 (1991). See also, B. Hurtig, "Pumping Gain: Understanding Dynamics Processors," *Electronic Musician*, **7**, no. 3, 56 (1991).

[3] G. W. McNally, "Dynamic Range Control of Digital Audio Signals," *J. Audio Eng. Soc.*, **32**, 316 (1984).

[4] Udo Zölzer, ed., *DAFX – Digital Audio Effects*, Wiley, Chichester, England, 2003.

[5] D. Giannoulis, M. Massberg, and J. D. Reiss. "Digital Dynamic Range Compressor Design—A Tutorial and Analysis," *J. Audio Eng. Soc.*, **60**, 399 (2012).

[6] "Dynamics Processors," ReneNote 155, Rane Corporation, 2005.

[7] S. Banerjee, "The Compression Handbook," 3d ed, Starkey Hearing Research & Technology, Starkey Laboratories, 2011.

Example Graphs

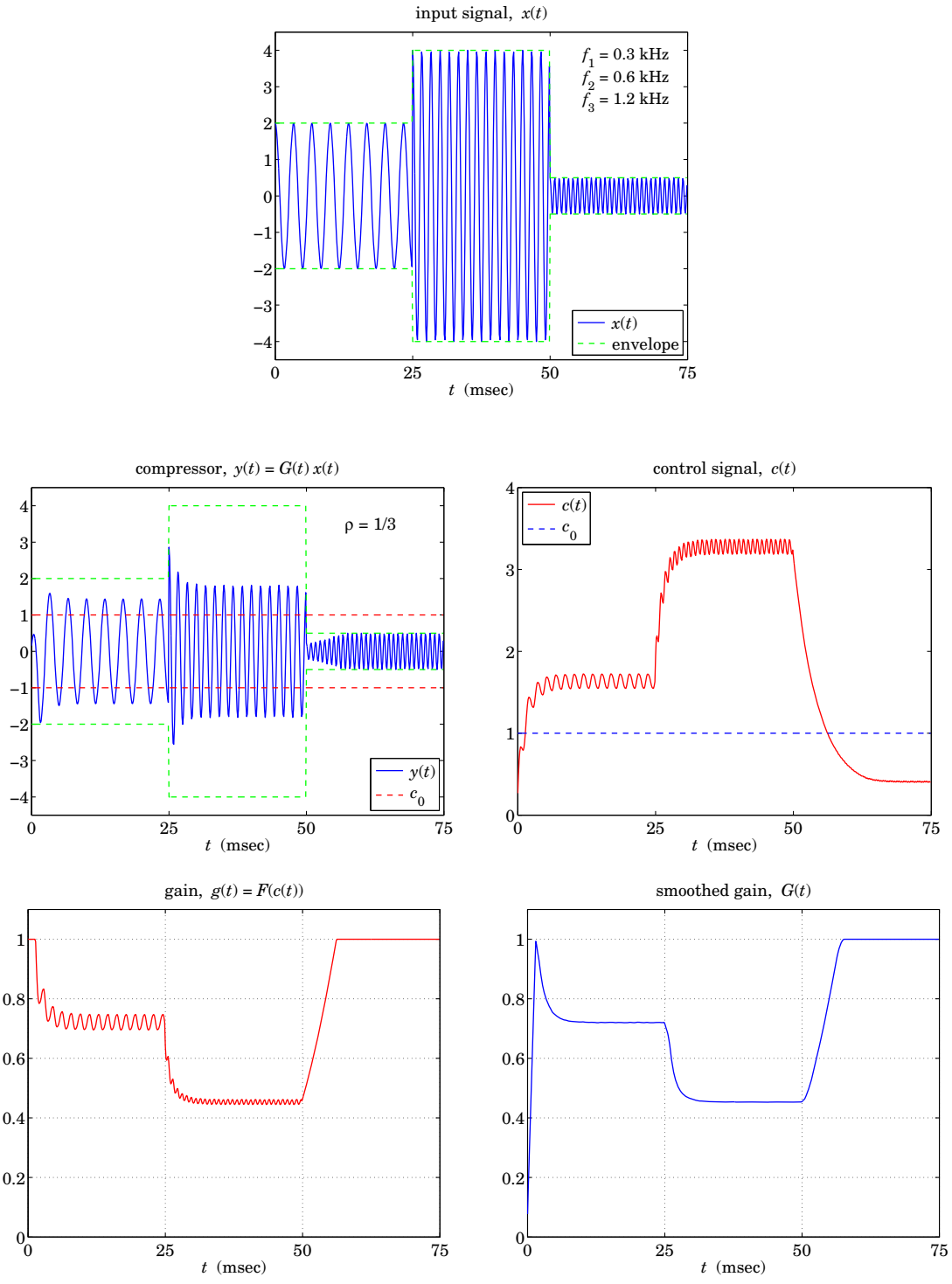

Fig. 5 Compressor, $\rho = 1/3$, $c_0 = 1$, using FIR smoother.

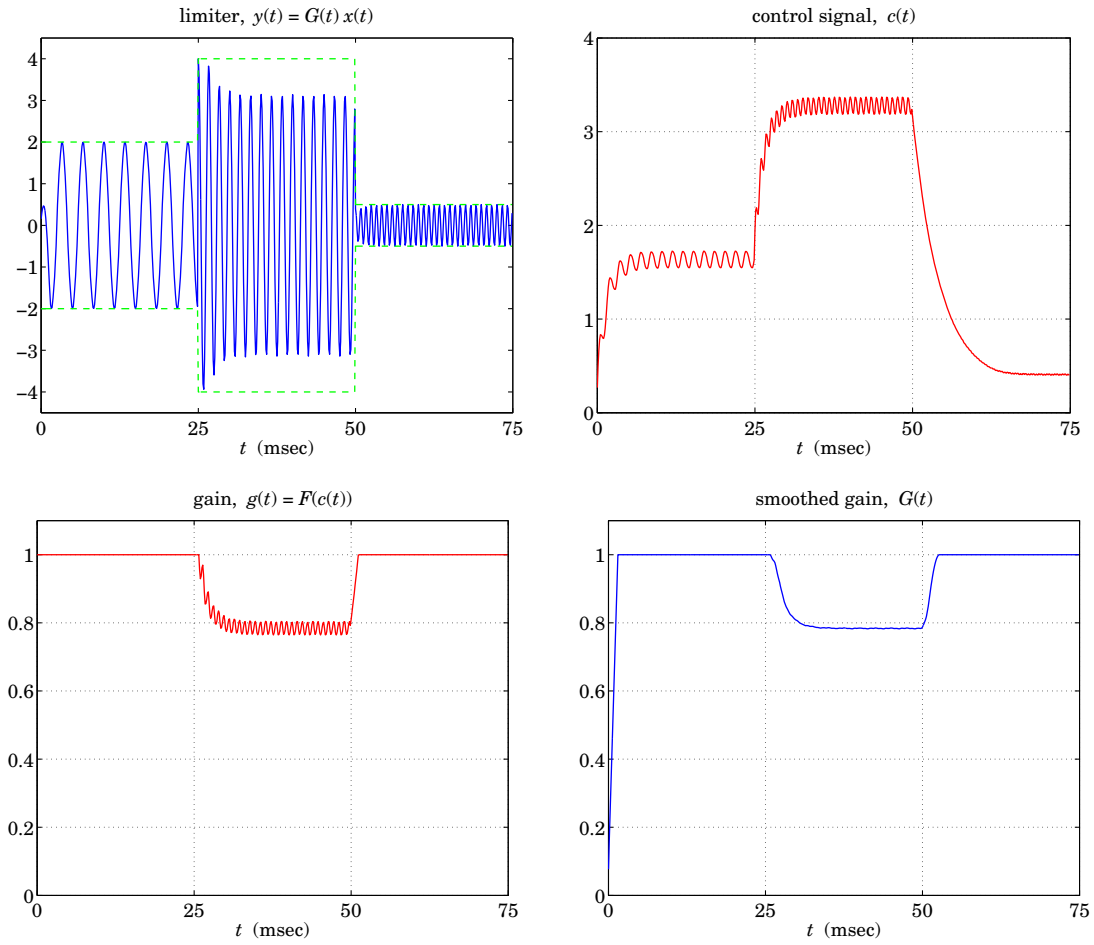

Fig. 6 Limiter.

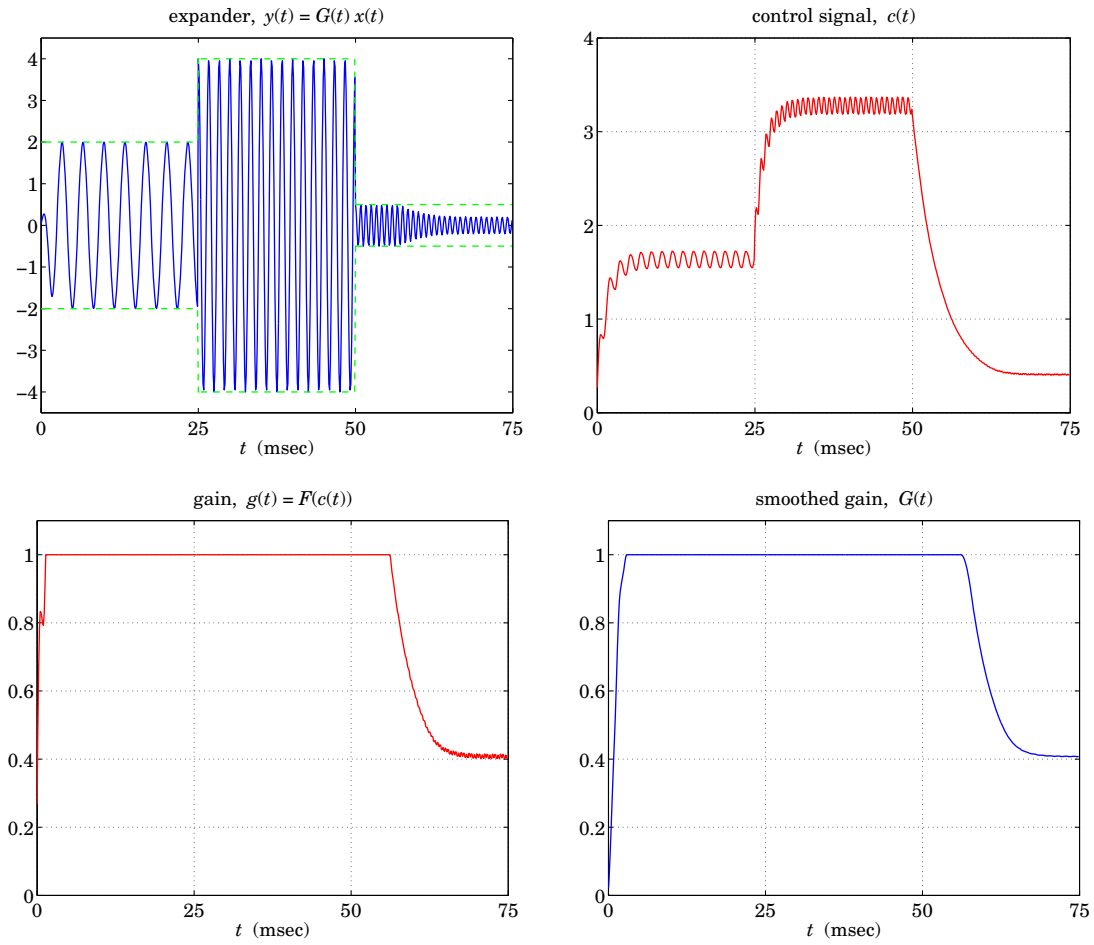

Fig. 7 Expander.

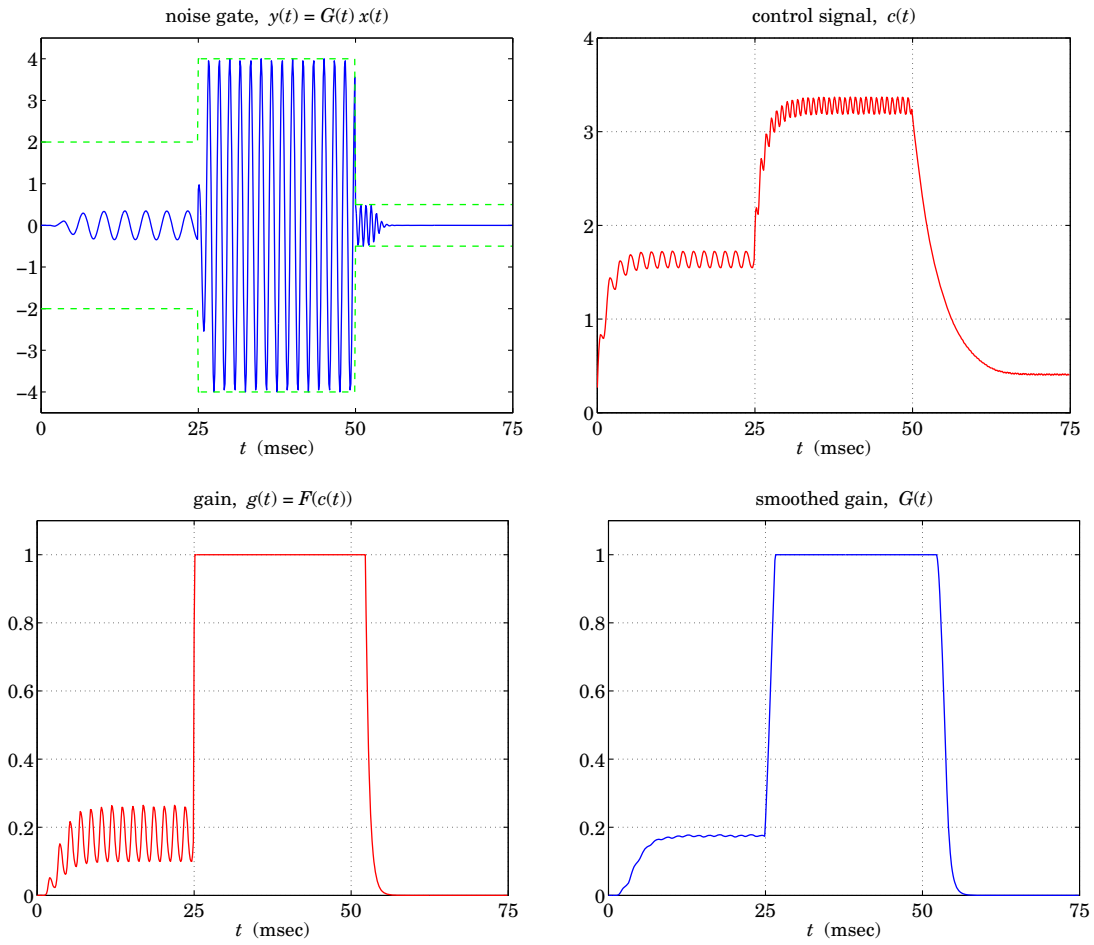

Fig. 8 Noise gate suppresing f_1 and f_3.

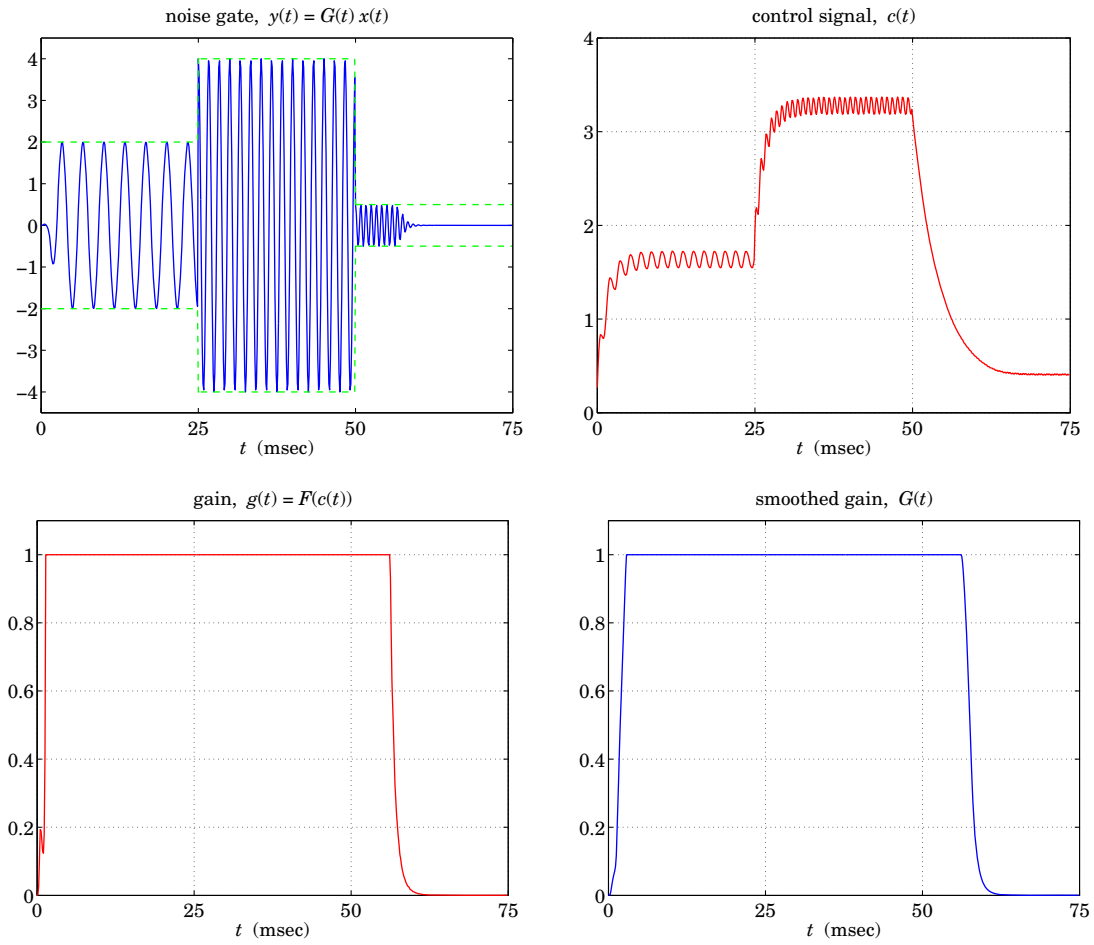

Fig. 9 Noise gate suppresing f_3 only.

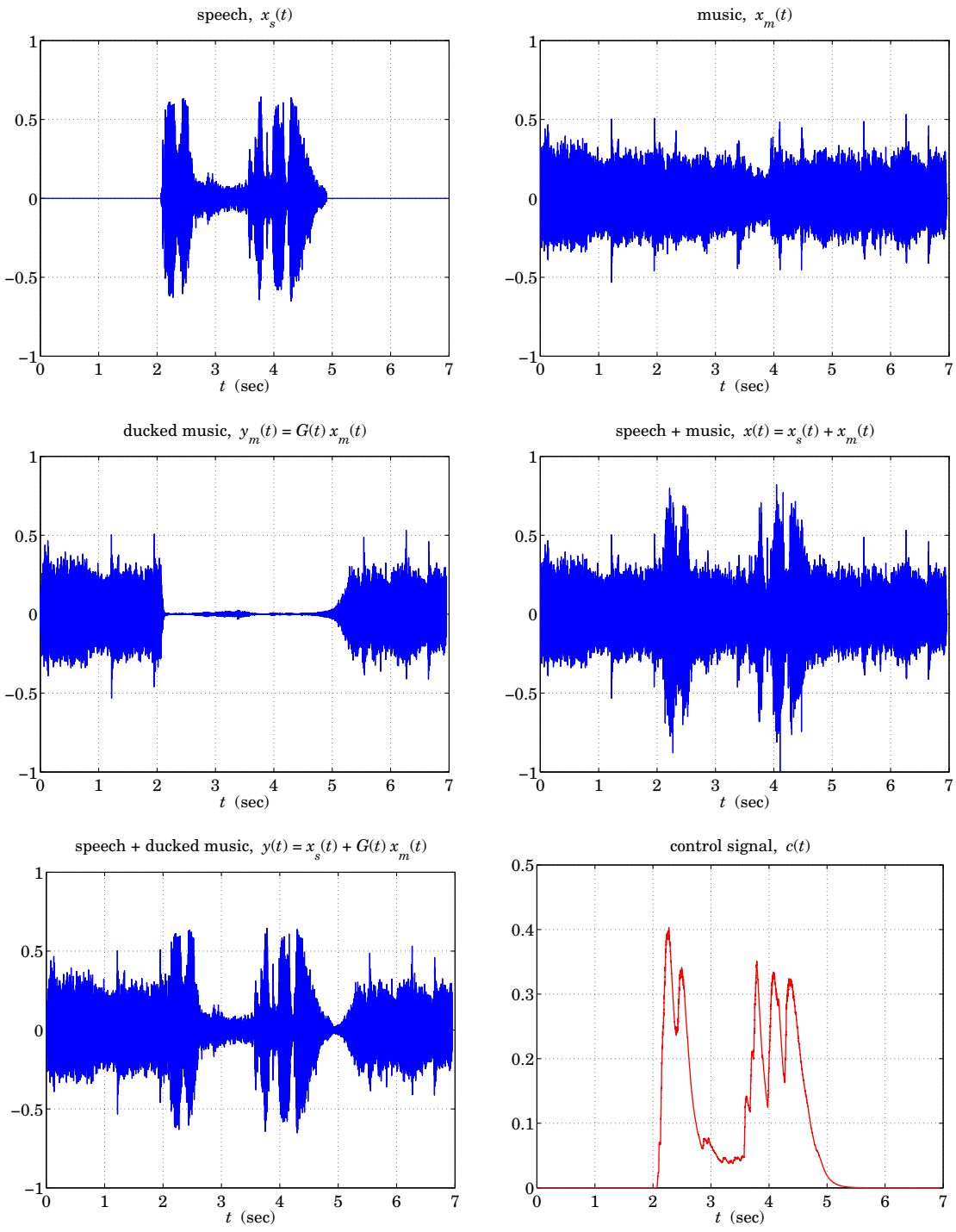

Fig. 10 Speech, music, speech + ducked music.

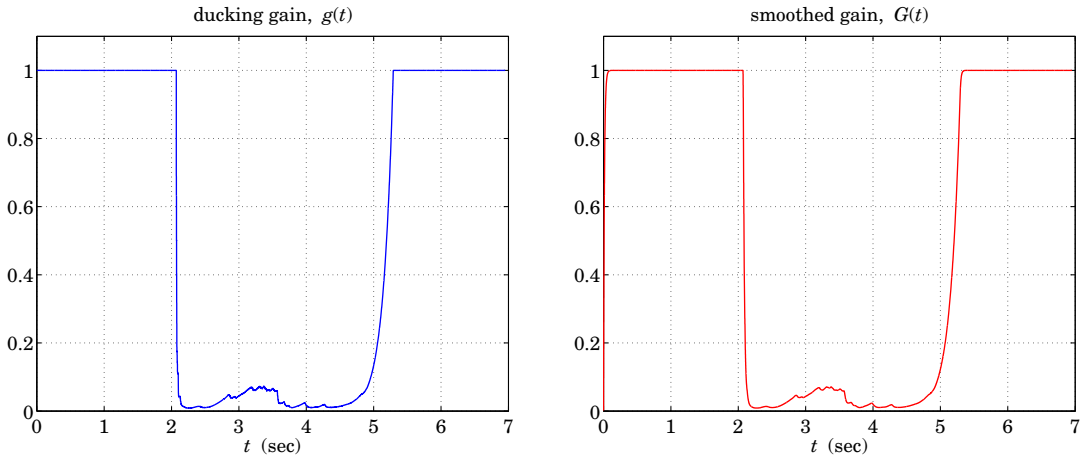

Fig. 11 Ducked music, control signal, and gains.