TECHNIUM

Technium Vol. 12, pp.15-22 (2023)
ISSN: 2668-778X
www.techniumscience.com

Recurrence relation for real-time audio fading

Lucian Lupşa-Tătaru

Faculty of Electrical Engineering and Computer Science,
"Transilvania" University of Braşov, Bd. Eroilor No. 29, Braşov, Romania,
lupsa@programmer.net, lucian.lupsa@unitbv.ro

Abstract. Starting from shaping the audio fade profile by an invertible rational function, with playback position as the input, we derive a recurrence relation that enables the audio volume updating as long as the time domain discretization is carried out with a constant step size. The resulted recurrence relation yields the audio volume current value as rational function of its previous value only, thus being suitable for efficiently implementing real-time adjustable fades whenever the adopted programming language (or software framework) allows the invocation of timing events. A JavaScript implementation, prepared for straightaway experimentation, and put forward in the paper, highlights the convenience of associating the derived recurrence relation with the "setInterval()" method in order to perform real-time audio fades, which can resemble either the logarithmic shape fade or the fade of exponential shape.

Keywords. Audio fade, adjustable fade, recurrence relation, rational function, timing events, real-time computing, programming techniques.

1. Introduction

In a broader sense, an audio fade designates either a strict increase or a strict decrease of the audio volume. Audio fades are applied in order to receive fade-up or fade-down effects and, more restrictive, fade-ins or fade-outs on the condition that one enforces a strict increase of the audio volume, starting from silence, or a strict decrease of the volume down to silence [1]-[5]. In the course of developing dedicated application software, the audio fades are traditionally implemented by making use of various transcendental functions (especially, exponential and logarithm) in order to depict the time-related evolution of the audio volume. Thus, the traditional manner of repeatedly updating the audio volume, during the fading process, involves intricate computations, being, therefore, more appropriate for off-line processing within media development applications geared towards audio production (e.g. audio editors).

Nevertheless, real-time audio capabilities are currently required by a fast increasing amount of application software, designed specifically to incorporate interactive audio [6]-[14]. In this regard, in order to improve the real-time audio experience, a rational function of degree 1, encompassing three coefficients, has previously been introduced in order to impose the time-related evolution of the audio volume during a fading process [15], [16]. It has also been validated that the employment of a rational function of degree 1 to customize the fading profile with a view to real-time processing, allows the implementation of adjustable fades (fade-up, fade-down) that can follow the pattern of the logarithmic fade or of the exponential shape fade [16].

The present investigation is directed towards figuring out a recurrence relation for audio volume updating with a view to implementing real-time adjustable fades. Assuming that the time domain

discretization is performed by means of a constant step size, it will be shown that the adoption of a degree 1 rational function to describe, during the fading process, the dependency of the audio volume upon the playback position, enables, eventually, the expressing of the audio volume current value in terms of its previous value only. More precisely, the audio volume value will come to be the return value of a rational function having as input just the previous value of the audio volume, whilst the coefficients of this rational function will depend on the size of time step, adopted for time domain discretization. It has to be pointed up that dealing with a recurrence relation to repeatedly update the audio volume, during the fading process, considerably facilitates the implementation as it is no longer necessary to use a method of counting the playback position within the fading effect. The proposed solution to audio fade constructing can be used in connection with programming languages or software frameworks that allow the calling of timing events [17]-[19].

2. Derivation of recurrence relation

One considers that the fade profile, depicting the time-related evolution of the audio volume, is built by means of the following rational function [15], [16]:

$$\begin{aligned} &v(\tau) = \frac{\tau - \alpha}{\beta\tau - \gamma}, \\ &\tau \in [0, \tau_f], \end{aligned} \tag{1}$$

wherein v designates the audio volume, τ is the playback position during the fading process, whilst τ_f represents the fade length. With $v_0 = v(0)$ as the audio volume occurring at the fade initiation, and $v_f = v(\tau_f)$ as the volume imposed at the end of the effect, the coefficients of (1) can be expressed as follows [16]:

$$\alpha = \frac{\tau_f v(0)}{v(0) + v(\tau_f) - \varepsilon^{-1} v(\tau_f)} = \frac{\tau_f v_0}{v_0 + v_f - \varepsilon^{-1} v_f}, \tag{2}$$

$$\beta = \frac{2 - \varepsilon^{-1}}{v(0) + v(\tau_f) - \varepsilon^{-1} v(\tau_f)} = \frac{2 - \varepsilon^{-1}}{v_0 + v_f - \varepsilon^{-1} v_f}, \tag{3}$$

$$\gamma = \frac{\tau_f}{v(0) + v(\tau_f) - \varepsilon^{-1} v(\tau_f)} = \frac{\tau_f}{v_0 + v_f - \varepsilon^{-1} v_f}, \tag{4}$$

where parameter ε, directly proportional to the playback position τ_μ at which the audio volume has its mean value μ_v, allows the fade profile customization. More precisely [16]:

$$\begin{aligned} &\varepsilon = \frac{\tau_\mu}{\tau_f}, \quad \varepsilon \in (0,1); \\ &v(\tau_\mu) = \mu_v = \frac{v(0) + v(\tau_f)}{2} = \frac{v_0 + v_f}{2}. \end{aligned} \tag{5}$$

It has been shown that the employment of rational function (1), wherein the coefficients are yielded by (2)-(4), enables the implementation of real-time audio fading by means of adjustable fades that can resemble the exponential shape fade or the fade of logarithmic shape, in accordance with the value of parameter ε in (2)-(4) [16].

One perceives that the use of generalized relation (1) to apply adjustable fades requires a method capable of returning the current playback time within the fading process. Thus, performing real-time audio fades via (1) is achievable by object-oriented programming, geared towards controlling the

audio content volume by counting the playback position from the initiation of the fading process. Nevertheless, more and more programming languages and software frameworks, like game engines, allow the employing of timing events, such as repeating the execution of statements by simply specifying the time interval between each two successive executions. In this context, it appears to be more convenient to receive the current value of the audio volume based on its previous value only. Hence, assuming that the time domain discretization is carried out with a constant step of size h, we aim at expressing the audio volume, occurring at playback position $\tau_{n+1} = \tau_n + h$, as a function of the audio volume received at playback position τ_n.

According to (1), at the particular playback position τ_n, the audio volume has the value

$$\begin{aligned} &v_n = v(\tau_n) = \frac{\tau_n - \alpha}{\beta\tau_n - \gamma}, \\ &\tau_n \in [0, \tau_f - h]. \end{aligned} \tag{6}$$

Therefore, since (1) is invertible,

$$\begin{aligned} &\tau_n = \frac{\gamma v_n - \alpha}{\beta v_n - 1}, \\ &\beta v_n \neq 1. \end{aligned} \tag{7}$$

Still relying on (1), to express the audio volume at playback position $\tau_{n+1} = \tau_n + h$ in terms of audio volume v_n, received at τ_n, one has to express τ_{n+1} as a function of v_n. Having in view (7), one identifies:

$$\tau_{n+1} = \tau_n + h = \frac{\gamma v_n - \alpha}{\beta v_n - 1} + h = \frac{(\beta h + \gamma)v_n - \alpha - h}{\beta v_n - 1} \tag{8}$$

and the corresponding audio volume, that is

$$\begin{aligned} v_{n+1} = v(\tau_{n+1}) = \frac{\tau_{n+1} - \alpha}{\beta\tau_{n+1} - \gamma} &= \frac{\frac{(\beta h+\gamma)v_n - \alpha - h}{\beta v_n - 1} - \alpha}{\beta\frac{(\beta h+\gamma)v_n - \alpha - h}{\beta v_n - 1} - \gamma} \\ &= \frac{(\beta h - \alpha\beta + \gamma)v_n - h}{\beta^2 h v_n - \beta h - \alpha\beta + \gamma}, \end{aligned} \tag{9}$$

where $\tau_{n+1} \equiv \tau_n + h \in [h, \tau_f]$.

Relation (9) highlights the audio volume at playback position τ_{n+1} as the return value of a rational function having as input just the audio volume at playback time τ_h, i.e.

$$v_{n+1} = \frac{a v_n - h}{b v_n - c}, \tag{10}$$

wherein the coefficients a, b, c depend on the original coefficients α, β and γ, which yield the fade shape, and step h, adopted here for time discretization:

$$a = \beta h - \alpha\beta + \gamma, \tag{11}$$

$$b = \beta^2 h, \tag{12}$$

$$c = \beta h + \alpha\beta - \gamma \tag{13}$$

Technium Vol. 12, pp.15-22 (2023)
ISSN: 2668-778X
www.techniumscience.com

or, taking into account (11),

$$c = 2\beta h - a\,. \tag{14}$$

Thus, the recurrence relation (10) allows the audio volume updating, having at hand the initial volume v_0, the final volume v_f, the audio fade length τ_f, the shape parameter ε, which is required for customizing the fade profile, and time step h.

3. The coefficients in the recurrence relation

During the fading effect of length τ_f, equation (10) yields the audio volume at the particular playback position τ_{n+1} as a function of the audio volume computed at playback position τ_n. It has to be pointed out that relation (10), wherein the coefficients are provided by (11)-(13), is valid as long as time discretization is accomplished by means of a constant time step of size h so that $\tau_{n+1} = \tau_n + h$, with $\tau_n \in [0, \tau_f - h]$ and $\tau_{n+1} \in [h, \tau_f]$.

Regardless of the implementation process, the values of the three coefficients in (10) have to be stored in global variables in order to be fetched by the function designed for volume automation. Compared to the coefficients in generalized relation (1), the coefficients in recurrence relation (10) additionally depend on time step h. Therefore, as the following JavaScript code indicates, a function designed to compute the coefficients of (10) should include five parameters, namely the fade length, the initial audio volume, the final audio volume, the parameter yielded by (5), used for fade shaping, and the size of the time step.

```
function evalCoeff( tauF, v0, vF, eps, stepH ) { /* computes the coefficients in (10) */
   /* tauF - the fade length
   v0 - initial audio volume
   vF - final audio volume
   eps - the parameter defined by (5)
   stepH - time step h */
   var commonToBetaGamma = v0 + vF - vF / eps;
      /* the denominator in expressions (2)-(4) of coefficients alpha, beta and gamma */
   var gamma = tauF / commonToBetaGamma;
   var alpha = gamma * v0;
      /* according to expressions (2) and (4) */
   var beta = ( 2.0 - 1.0 / eps ) / commonToBetaGamma;
      /* according to expression (3) */
   var beta_stepH = beta * stepH;
   coeffA = beta_stepH - alpha * beta + gamma;
      /* coefficient a in relation (10), as provided by (11); global scope */
   coeffB = beta * beta_stepH;
      /* coefficient b in relation (10), as provided by (12); global scope */
   coeffC = beta_stepH + beta_stepH - coeffA;
      /* coefficient c in relation (10), as provided by (14); global scope */
}
```

With global variables “coeffA”, “coeffB”, “coeffC” holding the values of the coefficients in (10), a JavaScript implementation of relation (10) is straightforward:

```
function vNp1( vN, stepH ) {
   /* implementation of relation (10) */
   var vol = ( coeffA * vN - stepH ) / ( coeffB * vN - coeffC );
   if ( vol < 0.0 ) { vol = 0.0; }
   else if ( vol > 1.0 ) { vol = 1.0; }
      /* the volume property of the audio object has to be within [0, 1] */
   return vol;
}
```

Technium Vol. 12, pp.15-22 (2023)
ISSN: 2668-778X
www.techniumscience.com

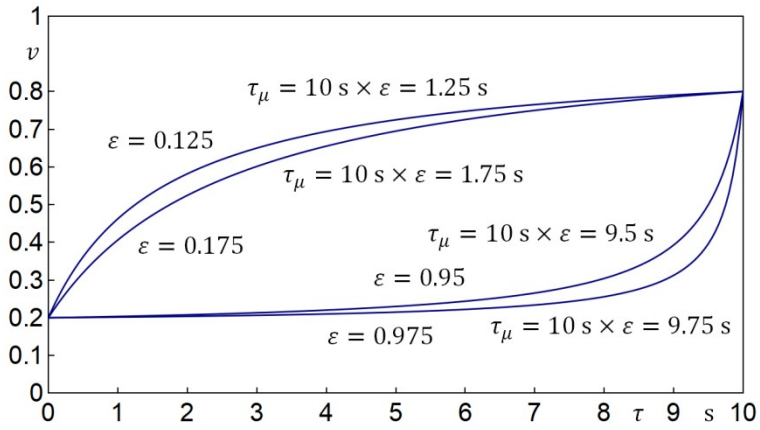

Fig. 1. Fade-up profiles received by calling functions "evalCoeff()" and "vNp1()" with the arguments: fade length of 10 s, initial volume of 0.2, final volume of 0.8, the shape parameter, provided by (5), in the set {0.125, 0.175, 0.95, 0.975}, time step of 0.05 s.

The four fade-up curves of Fig. 1 have been obtained by invoking functions "evalCoeff()" and "vNp1()", previously presented. The arguments utilized to call function "evalCoeff()" are as follows: fade length of 10 s, initial volume of 0.2, final volume of 0.8, the shape parameter ε, given by (5), in the set {0.125, 0.175, 0.95, 0.975}, and time step of 0.05 s. Accordingly, Table 1 indicates the return values of the coefficients in (10).

Table 1. The coefficients in (10) for parameter (5) in the set {0.125, 0.175, 0.95, 0.975}.

ε	a	b	c
0.125	-1.384773663	0.061728395	1.495884774
0.175	-2.1656	0.05408	2.2696
0.95	-12.366666667	1.8	12.966666667
0.975	-4.504081633	1.473469388	5.046938776

By carefully inspecting the audio fade profiles of Fig.1, one perceives that the curves obtained for parameter ε in {0.125, 0.175} follow the logarithmic fade shape whilst the curves corresponding to ε located in {0.95, 0.975} resemble the shape of exponential fade-ups. More specifically, according to formula (5), wherein $\tau_f = 10$ s, for $\varepsilon \in \{0.125,\ 0.175\}$, the playback time, τ_μ, at which the audio volume reaches the mean value of 0.5 is small enough to ensure a fast increase in volume in the beginning of the effect. On the other hand, for $\varepsilon \in \{0.95,\ 0.975\}$, the playback position at which the audio volume has the mean value of 0.5 is close enough to the fade-up length so that the rate of change of the audio volume comes to be much greater in the end of the effect than in the beginning of it.

4. Audio fading implementation

To validate the usefulness of recurrence relation (10) with the purpose of performing real-time fading, a plain JavaScript implementation has been prepared. For the sake of simplicity, just the fade profiles shown in Fig. 1 have been taken into account. Thus, the code is easily readable since the coefficients of relation (10) have already been computed and their values have been provided for each of the four shapes illustrated in Fig. 1. Within the implementation, one observes that the values of the coefficients in recurrence relation (10) result by calling function "selectCoeff()" with the appropriate argument that is the value selected for parameter "eps", which stands here for ratio yielded by (5). If parameter "eps" receives the value of 0.125 or the value of 0.175 then the fade-up effect will follow the pattern of the logarithmic fade while the playback position of reaching the audio volume mean value will be either 1.25 s or 1.75 s. On the other hand, if the argument of calling function "selectCoeff()" is equal to 0.95

Technium Vol. 12, pp.15-22 (2023)
ISSN: 2668-778X
www.techniumscience.com

then the resulted fade-up effect resembles the fade of exponential shape while the volume mean value corresponds to playback position of 9.5 s. Furthermore, if parameter “eps” receives a value that is not in the set {0.125, 0.175, 0.95} then the coefficients of (10) are implicitly correlated with a picture of resembling the exponential fade shape. This is carried out by simply imposing that the audio volume reaches the mean value of 0.5 at the playback position of 9.75 s. The implementation, simple enough for an immediate testing in any major browser, is given next.

```
<!DOCTYPE html>
<html>
<head>
   <title>Audio fading (recurrence relation)</title>
</head>
<body>
<script>

var coeffA, coeffB, coeffC; /* hold the values of coefficients in (10) */
var audElm, tmrId;

function selectCoeff( eps ) { /* assigns values to coeffA, coeffB, coeffC */
   switch( eps ) {
      case 0.125: // eps = 0.125
         coeffA = -1.384773663; coeffB = 0.061728395; coeffC = 1.495884774;
         break;
      case 0.175: // eps = 0.175
         coeffA = -2.1656; coeffB = 0.05408; coeffC = 2.2696;
         break;
      case 0.95:  // eps = 0.95
         coeffA = -12.366666667; coeffB = 1.8; coeffC = 12.966666667;
         break;
      default:    // eps = 0.975
         coeffA = -4.504081633; coeffB = 1.473469388; coeffC = 5.046938776;
   }
}
function vNp1( vN, stepH ) {
   /* plain implementation of relation (10) for volume in [0.2, 0.8] */
   var vol = ( coeffA * vN - stepH ) / ( coeffB * vN - coeffC );
   return vol;
}
function updateVol( vF, stepH ) {
   var vol = vNp1( audElm.volume, stepH );
   if ( vol < vF ) { audElm.volume = vol; }
   else {
      audElm.volume = vF;
      window.clearInterval( tmrId ); /* clears the timer to end the fading */
   }
}
function fadeUp() {
   audElm.currentTime = 0.0; audElm.volume = 0.2; /* initial condition */
   tmrId = window.setInterval( updateVol, 50, 0.8, 0.05 );
}
selectCoeff( 0.175 );
audElm = document.createElement( "AUDIO" );
audElm.src = "sample.mp3"; /* provided by user */
audElm.controls = true;
audElm.addEventListener( "play", fadeUp );
document.body.appendChild( audElm );

</script>
</body>
</html>
```

At a certain playback position, the audio volume is updated by invoking function “updateVol()”. The parameters “vF” and “stepH”, which are included in the definition of this function, designate the final volume and the step size, used for time domain discretization. The value received by parameter “vF” is needed to decide whether the fading process can be considered ended whilst the size of time step, received by parameter “stepH”, is passed to function “vNp1()” that implements the recurrence relation (10). It has to be emphasized that, since (10) does not incorporate the playback position, the block of code inside function “updateVol()” makes use of only the “volume” property of the audio object, that is, the use of “currentTime” property is not required within the code of this function.

To accomplish the fading process, function “fadeUp()” is invoked when the playback has started i.e. when the user has accessed the “Play” audio control. When called, function “fadeUp()” sets the initial volume to 0.2 and, via the “setInterval()” method of the “window” object, invokes function “updateVol()” once every 50 ms. As aforementioned, the arguments passed to function “updateVol()” represent the final volume that is 0.8 and the size of time step, in second, i.e. 0.05 s.

For the reason that function “selectCoeff()” is called with an argument of 0.175, the resulted fade-up effect resembles here the logarithmic shape fade corresponding to the situation, depicted in Fig. 1, in which the audio volume reaches the mean value of 0.5 at the playback position of 1.75 s.

5. Conclusion

Considering that the time domain discretization is accomplished by means of a constant step size, the present investigation advances a recurrence relation for updating the audio volume with the purpose of implementing real-time audio fades (fade-up, fade-down). The proposed recurrence relation boosts the effectiveness of implementation when used in conjunction with programming languages or software frameworks that provide access to timing events.

Derivation of the recurrence relation has been achievable due to the original manner of shaping the audio fade that is by means of a rational function of playback position. Since this rational function, initially adopted to describe the time-related evolution of the audio volume, is defined by the ratio of linear polynomials, the resulted recurrence relation provides the audio volume current value as rational function of just the previous value of the audio volume.

The audio fade profiles that can be received by implementing the suggested scheme are identical to those that can be obtained by using the rational function that has been initially employed to describe the audio volume as dependency on playback position. By running a plain JavaScript implementation, presented in the paper, one perceives that the attainable fade effects can act either as the logarithmic shape fade or as the exponential fade, in accordance with the value passed to a shape parameter, which is directly proportional to the playback position at which the audio volume will reach its mean value in the course of the fading process.

References

[1] S. LANGFORD: Digital Audio Editing. Correcting and Enhancing Audio in Pro Tools, Logic Pro, Cubase, and Studio One. Burlington, MA, USA, Focal Press, 2014.

[2] W. JACKSON: Digital Audio Editing Fundamentals. Get Started with Digital Audio Development and Distribution. Berkeley, CA, USA, Apress Media 2015. doi: 10.1007/978-1-4842-1648-4

[3] J. D. REISS, A. McPHERSON: Audio Effects. Theory, Implementation and Application. Boca Raton, FL, USA, CRC Press, 2015.

[4] C. SCHRODER: The Book of Audacity. Record, Edit, Mix, and Master with the Free Audio Editor. San Francisco, CA, USA, No Starch Press, 2011.

[5] The Audacity Team: Audacity(R) Free, Open Source, Cross-platform Audio Software. Audacity Manual, 2023. Adjustable Fade. https://manual.audacityteam.org/man/adjustable_fade.html

[6] R. L. BLEIDT et al.: Building the world’s most complex TV network: a test bed for broadcasting immersive and interactive audio. *SMPTE Motion Imaging Journal*, **126** (5), 26-34 (2017). doi: 10.5594/JMI.2017.2698618

[7] K. LIANG, B. SEO, A. KRYCZKA, R. ZIMMERMANN: IDM: An indirect dissemination mechanism for spatial voice interaction in networked virtual environments. *IEEE Transactions on Parallel and Distributed Systems*, **24** (2), 356-367 (2013). doi: 10.1109/TPDS.2012.91
[8] I. DEVLIN: HTML5 Multimedia. Develop and Design. Berkeley, CA, USA, Peachpit Press, 2012.
[9] N. HELYER, D. WOO, F. VERONESI: Artful media. The sonic nomadic: exploring mobile surround-sound interactions. *IEEE MultiMedia*, **16** (2), 12-15 (2009). doi: 10.1109/MMUL.2009.38
[10] K. KIM, J. SEO, S. BEACK, K. KANG, M. HAHN: Spatial audio object coding with two-step coding structure for interactive audio service. *IEEE Transactions on Multimedia*, **13** (6), 1208-1216 (2011). doi: 10.1109/TMM.2011.2168197
[11] E. LOELIGER, T. STOCKMAN: Wayfinding without visual cues: evaluation of an interactive audio map system. *Interacting with Computers*, **26** (5), 403-416 (2014). doi: 10.1093/iwc/iwt042
[12] L. ZENG, M. MIAO, G. WEBER: Interactive audio-haptic map explorer on a tactile display. *Interacting with Computers*, **27** (4), 413-429 (2015). doi: 10.1093/iwc/iwu006
[13] B. YU, J. HU, M. FUNK, R. H. LIANG, M. XUE, L. FEIJS: RESonance: lightweight, room-scale audio-visual biofeedback for immersive relaxation training. *IEEE Access*, **6**, 38336-38347 (2018). doi: 10.1109/ACCESS.2018.2853406
[14] M. SWEET: Writing Interactive Music for Video Games. A Composer's Guide. Upper Saddle River, NJ, USA, Addison-Wesley Professional, 2014.
[15] L. LUPSA-TATARU: Novel technique of customizing the audio fade-out shape. *Applied Computer Science*, **14** (3), 5-14 (2018). doi: 10.23743/acs-2018-17
[16] L. LUPSA-TATARU: Piecewise-defined function for effectively implementing the audio volume automation. *Technium: Romanian Journal of Applied Sciences and Technology*, **4** (9), 12-22 (2022). doi: 10.47577/technium.v4i9.7331
[17] A. BURNS, A. WELLINGS: Timing events and execution-time control. In: Concurrent and Real-Time Programming in Ada. Cambridge, Cambridge University Press, 2007, 361-390. doi: 10.1017/CBO9780511611230.016
[18] T. ABDELLATIF, J. COMBAZ, J. SIFAKIS: Rigorous implementation of real-time systems - from theory to application. *Mathematical Structures in Computer Science*, **23** (4), 882-914 (2013). doi: 10.1017/S096012951200028X
[19] D. FLANAGAN: JavaScript: The Definitive Guide, 7th Edition: Master the World's Most-Used Programming Language. Sebastopol, CA, USA, O'Reilly Media, 2020.