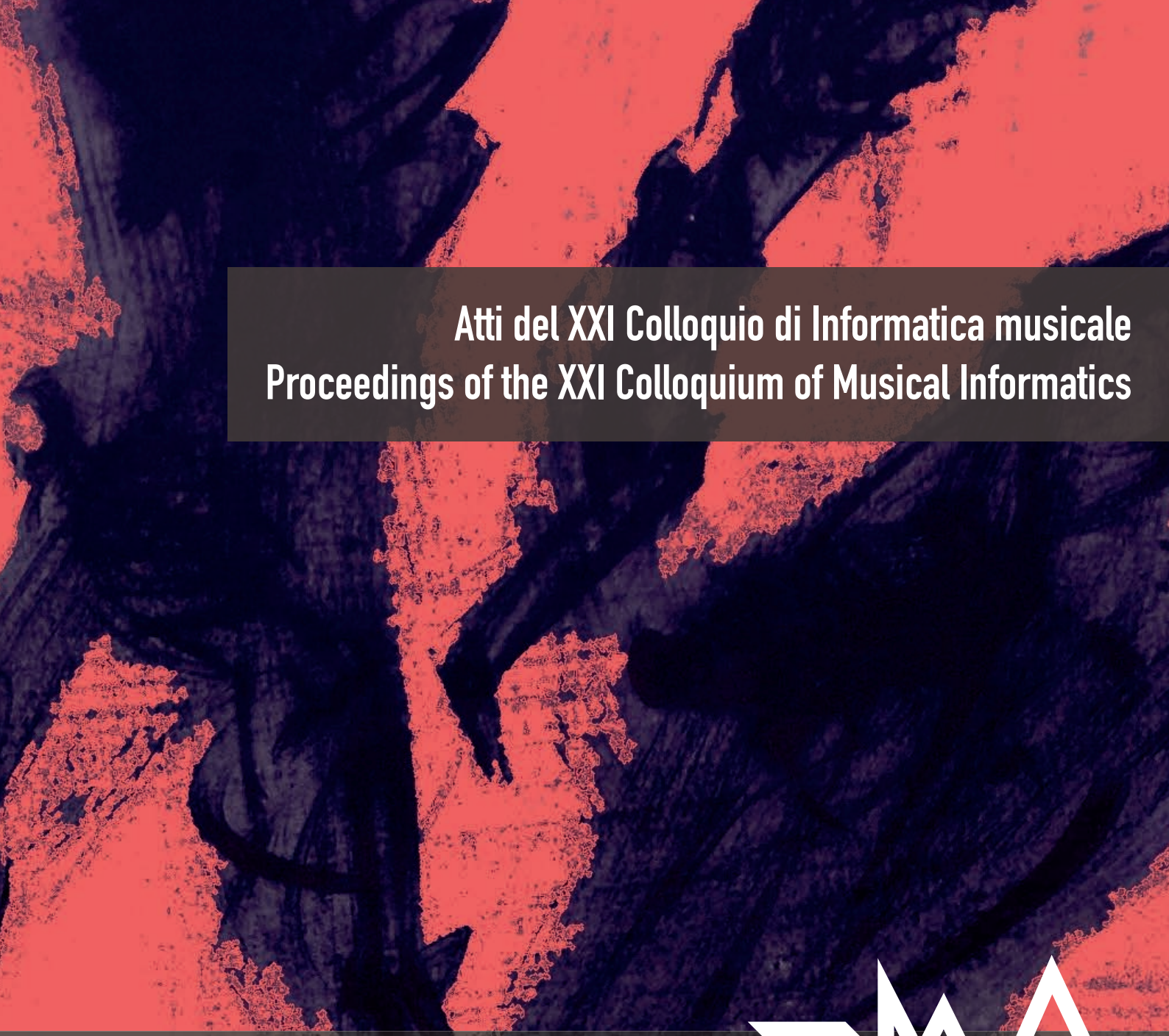

Atti del XXI Colloquio di Informatica musicale
Proceedings of the XXI Colloquium of Musical Informatics

XXI Colloquio di
Informatica Musicale

<Extending Interactivity>

Cagliari 28 settembre 1 ottobre 2016

AIMI
Associazione Informatica Musicale Italiana

SPAZIOMUSICA

« Extending interactivity »

Atti del XXI CIM - Colloquio di Informatica Musicale
Cagliari, 28 settembre–1 ottobre 2016

A cura di Anna Terzaroli e Andrea Valle

XXI CIM - Colloquio di Informatica Musicale
Cagliari, 28 settembre–1 ottobre 2016

AIMI – Associazione di Informatica Musicale Italiana - www.aimi-musica.org
Spaziomusica – www.spaziomusicaproject.com

Con il contributo di:
Regione Autonoma della Sardegna, Assessorato della Pubblica Istruzione, Beni Culturali, Informazione, Spettacolo e Sport
Comune di Cagliari, Assessorato alla Cultura, Pubblica Istruzione, Sport, Spettacolo e Politiche Giovanili

Con la collaborazione di:
Università di Cagliari, Dipartimento di Storia, Beni Culturali e Territorio
Conservatorio Statale di Musica Giovanni "Pierluigi da Palestrina" di Cagliari
Labimus
10 Nodi - I festival d'autunno a Cagliari

Atti del XXI CIM - Colloquio di Informatica Musicale
A cura di Anna Terzaroli e Andrea Valle
ISBN: 9788890341342
sito: http://www.aimi-musica.org/

Riferimento BibTeX:

```
@proceedings{XVIIICIM,
    Editor = {Anna Terzaroli and Andrea Valle},
    Organization = {AIMI - Associazione Informatica Musicale Italiana},
    Publisher = {DADI - Dip. Arti e Design Industriale. Università IUAV di Venezia},
    Title = {Extending interactivity. Atti del XXI CIM - Colloquio di Informatica Musicale},
    Year = {2016}}
```

Copyright

These proceedings, and all the papers included in it, are an open-access publication distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author(s) and source are credited. To view a copy of this license, visit http://creativecommons.org/licenses/by/3.0/ or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

Typeset with ConTEXt by Andrea Valle

INTERPOLAZIONI EASING: IMPLEMENTAZIONE E APPLICAZIONI AUDIO

Renato Messina
ISSM Vincenzo Bellini, Catania
messinarenato@gmail.com

SOMMARIO

L'estensione delle tecniche della computer animation ad ambiti seriali affini, come quello della computer grafica o del web design, ha determinato un'ampia diffusione delle funzioni easing di interpolazione. La stilizzazione prodotta da queste curve di transizione, definite sulla base di modelli cinematici caratteristici, suggerisce in ambito audio strutture simboliche ed iconiche efficaci nella progettazione di segnali acustici e nella comunicazione di significati espressivi correlabili a schemi isomorfici gestuali. Nell'articolo viene descritta l'implementazione Java di un external multipiattaforma per il proxy mxj [1] di Max[1] e PD[2], dotato di una libreria di 30 curve e rivolto allo sviluppo di architetture cross-modali e di display uditivo.

1. INTRODUZIONE

Nella computer animation le interpolazioni easing forniscono un'interfaccia di programmazione ad alto livello con cui assegnare una curva di accelerazione ad un oggetto in movimento. Le variazioni di velocità che in tal modo ne definiscono lo spostamento rendono le transizioni più fluide e realistiche attraverso la riproduzione di tipologie cinematiche proprie del moto dei corpi puntiformi o della biomeccanica. Per estensione, tali interpolazioni, chiamate anche tweening, possono applicarsi, oltre che alla posizione (motion tweening), anche a parametri descrittivi dell'oggetto (shape tweening), come dimensione e forma, indirettamente correlabili ad altre grandezze cinematiche fondamentali, quali massa e peso.

In via generale, se le easing esprimono una relazione in cui il dominio della funzione è il tempo e il codominio è l'insieme dei valori assunti dall'argomento che si vuole animare, la presenza di una linea temporale di accelerazione comune garantisce potenzialmente un'integrazione coerente automatica ad un numero indefinito di codomini rappresentativi del set di avars (animation variables) impiegato. Operazioni di inversione e rovesciamento applicate alle rampe di singoli parametri riescono, così, a creare effetti complessi di interazione dinamica tra le proprietà strutturali interne, la traiettoria e la posizione dell'oggetto animato. Un'analoga ricchezza di relazioni, in ambito musicale, si può evidenziare nel raggruppamento di vincoli metrici, agogici e dinamici presente nelle tradizionali tecniche del 'legato' e nelle prassi esecutive di appoggiature, glissandi e portamenti[3].

Copyright: © 2016 Renato Messina. This is an open-access article distributed under the terms of the Creative Commons Attribution License 3.0 Unported, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

2. NOMENCLATURA E CLASSI

La nomenclatura di una classe easing individua attraverso un prefisso e una radice i due elementi formali che la caratterizzano. Il prefisso specifica una delle tre versioni, *In*, *Out*, *InOut,* del profilo di accelerazione esponenziale, logaritmico o ibrido della funzione richiamata. La caratteristica delle prime due versioni, *In* e *Out*, è l'angolo di discontinuità presente nella fase conclusiva, o iniziale, della curva, vedasi figura 1. I contenuti espressivi e mimici che tale peculiarità riesce a veicolare sono valutabili, in termini di efficacia, in rapporto al contesto di applicazione. Ad esempio, da un punto di vista cinematico, gli inviluppi logaritmici prodotti dalla versione *Out* possono descrivere forze di tipo esplosivo, con attacco percussivo, mentre gli inviluppi esponenziali della versione *In*, sono più adatti a descrivere forze reattive, con un'accelerazione progressiva che culmina in un arresto immediato, come avviene nell'esecuzione di salti o di lanci. La versione *In*, d'altro canto, se considerata da un punto di vista pliometrico, può risultare innaturale per l'assenza di vibrazioni residue e per il suo coefficiente di restituzione nullo; e, viceversa, può essere consistente nel caso di rappresentazioni metaforiche che simulino sistemi elettronici a controllo numerico. Per il senso di precisione che comunica nella sua fase di arresto, si potrebbe anche dire che la versione *In* esprima in modo paradigmatico l'effetto di applicazione di uno schema di controllo feedforward (anticipativo) delle forze necessarie al sistema per l'esecuzione di un determinato piano motorio [2]. L'ultima delle tre versioni, identificata dal prefisso *InOut,* specifica un andamento *In* nella prima

[1] www.cycling74.com.

[2] https://puredata.info.

[3] L'applicazione delle curve all'articolazione melodica evidenzia una correlazione tra sistemi discreti e sistemi continui in grado di fornire un'interessante descrizione della linea melodica come successione di curve strutturali di transizione ancor prima che come sequenza di intervalli non interpolati.

metà della curva e *Out* nella seconda metà, dunque con attacco e release morbidi. La cosiddetta funzione S-curve, una cubica che implementa questa forma di andamento, è tra le più frequentemente usate, sia in contesti grafici che meccatronici, proprio per la fluidità che conferisce al movimento. È interessante rilevare come la diffusione della versione *InOut* trovi un riscontro biomeccanico in diversi modelli in cui il movimento è scomposto nelle sue due fasi di accelerazione e decelerazione. Ad esempio, il modello a due componenti di Woodworth [3], o il profilo di velocità a campana [4], o il modello 'minimum-jerk', secondo cui uno degli obiettivi del sistema nervoso è proprio la massimizzazione della 'dolcezza' (smoothness) del movimento [5].

Se il prefisso del nome della classe indica l'andamento esponenziale o logaritmico dell'accelerazione, la radice ne specifica la funzione. Le librerie ne implementano circa 30, riassumibili in 3 categorie.

1) Esponenziali, dove il nome della classe indica l'esponente assegnato per calcolare la pendenza della funzione.
2) Sinusoidale e Circolare.
3) Back, Elastic e Bounce, le quali, essendo funzioni non monotone, posseggono un andamento più articolato e creano effetti di anticipazione o rimbalzo con una verosimiglianza sufficientemente realistica. Vedasi in Tabella 1 il riassunto delle classi.

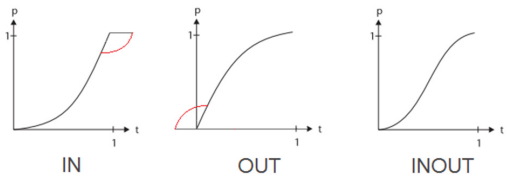

Figura 1. Angolo di discontinuità delle versioni *In* e *Out*. In ascissa è rappresentato il tempo, in ordinata la posizione del parametro associato.

Versioni			Funzione
In	*Out*	*InOut*	Sine
In	*Out*	*InOut*	Quadratic
In	*Out*	*InOut*	Cubic
In	*Out*	*InOut*	Quartic
In	*Out*	*InOut*	Quintic
In	*Out*	*InOut*	Exponential
In	*Out*	*InOut*	Circular
In	*Out*	*InOut*	Back
In	*Out*	*InOut*	Elastic
In	*Out*	*InOut*	Bounce

Tabella 1. Nomenclatura delle classi easing elencate in ordine di pendenza e complessità delle funzioni.

3. FUNZIONI

3.1 Esponenziali

Le curve esponenziali sono rese con le relative funzioni distinte in base al numero dell'esponente:

OutLinear $f(t) = t^1$ (1)
OutQuadratic $f(t) = t^2$ (2)
OutCubic $f(t) = t^3$ (3)
OutQuartic $f(t) = t^4$ (4)
OutQuintic $f(t) = t^5$ (5)
OutExponential $f(t) = t^{10}$ (6)

Le versioni *In* sono ottenute tramite flip orizzontale e verticale delle versioni *Out*. Ad esempio la funzione InCubic assumerà la forma: $f(t) = (t-1)^3 + 1$.

Le versioni *InOut*, come già detto, sono ricavate attraverso un metodo che applica la versione *In* alla prima metà della rampa e quella *Out* alla seconda. Si veda ad esempio una codifica della curva InOutCubic, in Java:

```
public double easeInOutCubic (double value){
double v = 2.0d * value;
if (v < 1.0d){
return 0.5d * Math.pow(v, 3.0d);
}
v -= 2.0d;
return 0.5d * (Math.pow(v, 3.0d) + 2.0d);
}
```

Oppure, in Max (figura 2), la stessa curva, generata all'interno di un buffer di 512 campioni, con frequenza di campionamento 44.1 kHz.

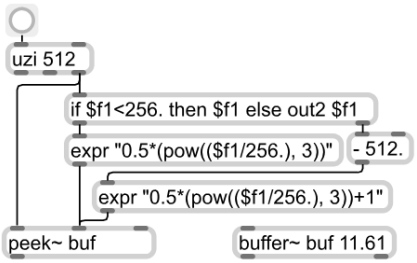

Figura 2. Curva InOutCubic (S-curve) in Max.

Nelle funzioni esponenziali, l'incremento dell'ordine di potenza produce un effetto di intensificazione della drammaticità, o reattività, della rampa, in contrapposizione ad un effetto di moto inerziale tipico delle rampe con esponenti più bassi.

3.2 Sinusoidale e circolare

La curva sinusoidale è calcolata con $\pi/2$ radianti di una cosinusoide (*In*) o di una sinusoide (*Out*):

SineOut $f(t) = \sin(t * \pi/2)$ (7)
SineIn $f(t) = -1 * \cos(t * \pi/2) + 1$ (8)

Esibisce dunque un andamento molto simile alla quadratica ma con una pendenza leggermente inferiore.

La circolare è calcolata con l'equazione della circonferenza goniometrica: $x^2 + y^2 = 1$, che per $\pi/2$ assume la forma:

$y = \sqrt{1-(x-1)^2}$

Le prime due versioni, *In* e *Out* sono:

easeInCirc $f(t) = -1(\sqrt{1-t^2}-1)$ (9)

easeOutCirc $f(t) = \sqrt{1-(t-1)^2}$ (10)

Presenta un livello di pendenza collocabile tra la Quintica e l'Esponenziale. È la funzione che rende l'effetto di rallentando o accelerando più omogeneo per la distribuzione regolare delle variazioni di velocità sull'intera fase.

3.3 Back

La Back deriva il suo nome dalla versione *In*, dove si osserva un superamento backwards molto morbido della soglia iniziale. È realizzata con una cubica con un overshooting del 10% circa gestito dal fattore 'g' dell'equazione seguente [6]:

InBack $f(t) = (1+g)t^3 - gt^2$ (11)

OutBack $f(t) = 1 - backIn(1-t)$ (12)

3.4 Elastic

La classe Elastic si basa su 2 funzioni distinte; la prima produce 3 cicli sinusoidali, la seconda traccia la curva decrescente che crea l'effetto di smorzamento.

OutElastic $f(t) = \sin(6\pi)\, 2^{10}(t-1)$ (13)

La *In*, viceversa, simula l'effetto di un progressivo accumulo di energia cinetica seguito da una rapida fase di attacco. Può dunque esemplificare fenomeni di discontinuità, o turbolenza, precedenti alla stabilizzazione di un dato livello.

3.5 Bounce

La classe Bounce produce l'effetto di un rimbalzo. Esegue 4 differenti funzioni in rapporto alla progressione della fase [6]. A titolo esemplificativo, si riporta la versione Out codificata in Java[4]:

```
if (p < 1 / 2.75) { // p < 36%
  return 7.5625 * p * p;
}
else if (p < 2 / 2.75) { // p < 72%
  return 7.5625 * (p -= 1.5 / 2.75) * p +
  0.75;
// curva esponenziale con inversione al 50%
}
else if (p < 2.5 / 2.75) { // p < 90%
  return 7.5625 * (p -= 2.25 / 2.75) * p +
  0.9375;
// curva esponenziale con inversione al 81%
}
  return 7.5625 * (p -= 2.625 / 2.75) * p
  + 0.984375;
// curva esponenziale con inversione al 95%
```

[4] https://github.com/greensock/GreenSock-JS/blob/master/src/uncompressed/easing/EasePack.js.

4. PARAMETRI

In molte librerie easing, ad esempio in quelle presenti nei programmi di grafica vettoriale Adobe, per il calcolo delle funzioni vengono adoperati 4 parametri: t (time), b (begin), c (change), d (duration) [7]. Il parametro 'time' indica la suddivisione temporale con cui la durata complessiva 'duration' della rampa viene misurata. Il parametro 'change' indica il valore di destinazione da raggiungere partendo dal valore offset 'begin'. La formula dell'interpolazione lineare ivi applicata,

$$i = c * t/d + b \quad (14)$$

distribuisce la rampa (*c*) lungo l'arco di una durata prestabilita. Il rapporto time/duration (*t/d*) è la fase normalizzata della rampa eseguita alla frequenza controllata dalla suddivisione temporale adottata ('time'). Le classi effettuano dunque una scansione temporale del parametro 'change' a partire dal valore 'begin'. La funzione con cui viene calcolata la curva si applica alla fase suddetta (time/duration).

Per creare una rampa lineare si userà dunque la funzione:

```
Math.linearTween = function (t, b, c, d) {
       return c*t/d + b;
}.
```

Esempio di valore restituito da una funzione easeInQuadratic:

```
Math.easeInQuad = function (t, b, c, d) {
       return c*(t/=d)*t + b;
}.
```

Una semplificazione presente in molte librerie consiste nell'omettere i 2 parametri 'change' e 'begin' (che vengono calcolati successivamente sui valori restituiti dalla funzione) e di gestire la rampa come semplice coefficiente di fase.

5. IMPLEMENTAZIONE DELL'EXTERNAL "EASING"

Per una maggiore usabilità delle interpolazioni easing in ambito audio, è stato programmato l'external Java "easing", eseguibile in ambiente PD e Max tramite il proxy mxj. La versione attuale della classe si basa su un blocco di funzioni predefinite, riassunto nella tabella 1, scritto da Michael Heuer [8] e tratto da una delle librerie più diffuse, sviluppata da Robert Penner [6]. L'implementazione si basa, inoltre, su una ricodifica dell'opcode

"line"[5] di Max finalizzata ad un uso dell'external più immediato ed efficiente.

Il clock interno dell'external, nella versione attuale, è impostato in modo costante a 1 ms e fornisce il parametro 'time' delle interpolazioni, ovvero la suddivisione temporale interna con cui la 'duration' viene misurata. La formula applicata per la ricodifica di "line" è:

$$result = begin + (time/(((1/(change - begin))) * duration)) \quad (15)$$

La caratteristica principale di "line" è l'assegnazione '*begin = result*' che viene aggiornata ad ogni nuovo avvio di rampa, congiungendo tra loro i segmenti di interpolazioni successive. Questa particolarità consente variazioni continue in runtime all'interno dell'inviluppo in esecuzione e rende l'algoritmo adatto ad un uso in tempo reale. In casi limite, quando i valori in ingresso si succedono con una rapidità superiore al tempo assegnato al fattore 'duration', i valori in uscita tendono alla successione stessa in ingresso. La rampa segue, dunque, l'andamento della funzione solo per la durata relativa all'assenza di nuovi dati in ingresso. Ad esempio, per una funzione *InOut* impostata con una durata di 1000 ms, se i valori in ingresso si susseguono a intervalli non superiori ai 500 ms, verrà letta solo la porzione *In*.

5.1 Algoritmo

Per la realizzazione delle curve, l'external implementa il seguente algoritmo:

0) Quando l'external riceve un valore in ingresso, esegue un metodo che preleva il valore corrente in uscita (*outNow*): *outStart = outNow.*

1) Calcola la differenza (*diff*) tra il valore in ingresso e il valore in uscita prelevato al punto precedente: *diff = change - outNow*.

2) Avvia la rampa e ne calcola la fase corrente, iterativamente per l'intero svolgimento della durata assegnata, come differenza tra i valori in uscita (*outFinal*) e il valore precedentemente prelevato al punto 0: *running = outFinal - diff*.

3) Converte la fase corrente in valori normalizzati: *running normalized = running/diff*.

4) La fase corrente normalizzata invoca la classe di interpolazione scelta dall'utente: *f(out) = running normalized*.

La selezione delle funzioni avviene in runtime tramite le API Reflection di Java. Il metodo che richiama la classe di interpolazione viene quindi lanciato solo quando se ne modifica l'istanza con un conseguente risparmio di risorse.

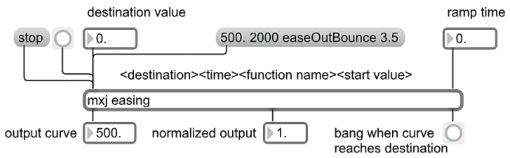

Figura 3. Interfaccia dell'external.

[5] https://docs.cycling74.com/max5/refpages/max-ref/line.html.

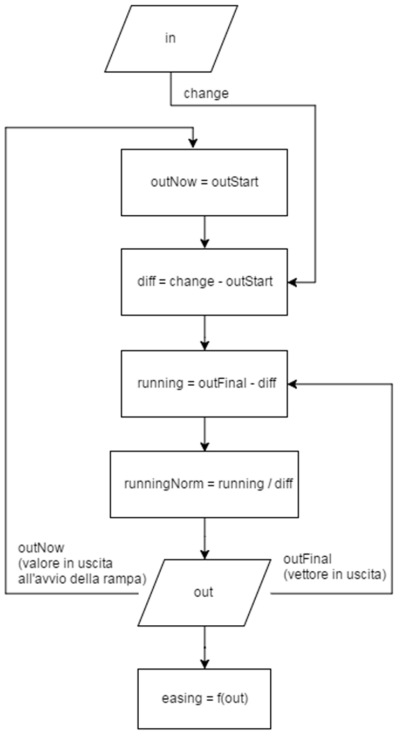

Figura 4. Diagramma di flusso dell'algoritmo.

6. INTERFACCIA

L'external possiede 2 inlet e 3 outlet (figura 3). Gli argomenti accettati sono: 'nome della funzione' e 'change' (inlet sinistro), 'duration' (inlet destro).

Per il settaggio degli argomenti tramite messaggi composti, la sintassi è:

<nome funzione> <change> <duration> <start point (opzionale)>.

Le proprietà implementate richiamabili tramite messaggi inviati all'inlet sinistro sono:

- la selezione della funzione che si desidera applicare <nome funzione>.
- l'interruzione dell'esecuzione della rampa sul suo valore corrente <stop>.
- l'esecuzione della sola rampa normalizzata, elaborata sulla base della durata impostata <bang>.

Gli outlet restituiscono in uscita:

- la rampa con i valori pesati (outlet sinistra);
- la rampa normalizzata (outlet centrale);
- un bang a conclusione della rampa (outlet destro).

I metodi che gestiscono le variabili numeriche in ingresso accettano, in overloading, sia tipi numerici interi (int) che in virgola mobile (float).

7. APPLICAZIONI

La sperimentazione qui condotta sull'uso delle interpolazioni easing in ambito audio ha analizzato principalmente l'applicazione delle curve a inviluppi di ampiezza e di frequenza nel dominio del tempo. In questo contesto le easing dimostrano rilevanza soprattutto nel trattamento di spettri estesi ed inarmonici, atti a simulare fenomeni cinetici prodotti da reazioni vincolari, come attrito e sfregamento, o azioni fluidodinamiche [9], in cui l'oggetto sonoro risultante tende a riflettere più l'intenzione, e la rapidità, con cui è stato generato che il significato 'melodico' correlato. L'aspetto di maggior interesse delle easing sembra risiedere, in tal senso, nella funzione isomorfica di stilizzazione e di rappresentazione iconica che esse svolgono nei confronti di codici espressivi figurativi [10]. Per l'immediatezza con cui le primitive di movimento, e il clustering dei parametri che esse organizzano, risultano assimilabili a stereotipi cinesici e gestuali riconducibili a categorie mimiche e prossemiche [11].

7.1 Relazioni cross-modali

Nell'esempio proposto[6] per testare alcune delle possibili applicazioni musicali dell'external, si è usato come sorgente audio un generatore di rumore rosa filtrato con un passa banda e un passa alto controllati da funzioni easing; le medesime curve di interpolazione sono state applicate contemporaneamente sia al fattore Q che alla frequenza centrale e di taglio del filtro [12]. Per la scrittura delle versioni *InOut* la rampa è stata scomposta in 2 parti; nella seconda parte l'andamento è invertito al fine di compensare la discontinuità dell'angolo di decadimento illustrato in figura 1.

In ambito cinematico, l'ordinata della funzione controlla rampe d'ampiezza e rampe di traiettoria: le prime sono applicabili alle grandezze di massa e peso (afferenti alla pressione esercitata nei fenomeni di frizione), le seconde allo spostamento spaziale del corpo. Parallelamente in ambito audio, le rampe di frizione definiranno l'inviluppo d'ampiezza, così determinandosi un incremento figurato di massa proporzionale alla velocità delle fasi ADSR; le rampe di traiettoria definiranno invece le curve di portamento, con l'accelerazione dello spostamento spaziale proporzionale allo shifting in frequenza dello spettro audio.

8. BIBLIOGRAFIA

[1] Topher La Fata: *Writing Max Externals in Java*, https://pcm.peabody.jhu.edu/~gwright/stdmp/docs/writingmaxexternalsinjava.pdf.

[2] P. Morasso e V. Sanguineti: *Paradigmi di Controllo Motorio*, Dipartimento di Informatica, Sistemistica, Telematica (DIST), Università di Genova.

[3] R.S. Woodworth: *The accuracy of voluntary movement*. Psychological Review Monograph Supplements, 3, no. 3, 1899.

[4] P. Morasso: "Spatial control of arm movements," *Experimental Brain Research*, 42, pp. 223-227, 1981.

[5] T. Flash and N. Hogan: "The coordination of arm movements: an experimentally confirmed mathematical model," *Journal of Neuroscience*, 7, pp. 1688-1703, 1985.

[6] Rober Penner: *Robert Penner's Programming Macromedia Flash MX*, McGraw-Hill / OsborneMedia, 2002.

[7] Oswald Campesato: *jQuery, CSS3, and HTML5 for Mobile and Desktop Devices*, Mercury Learning & Information, 2013.

[8] Michael Heuer: *Interface Easing Function*, http://www.dishevelled.org/interpolate/apidocs/org/dishevelled/interpolate/EasingFunction.html.

[9] F. Avanzini, S. Serafin, and D. Rocchesso: "Interactive simulation of rigid body interaction with friction-induced sound generation," *IEEE Trans. Speech and Audio Processing*, Vol. 13, No. 5, pp. 1073–1081, 2005.

[10] M.M. Blattner, D.A. Sumikawa, and R.M. Greenberg: "Earcons and icons: their structure and common design principles," *Human-Computer Interaction*, Vol. 4, No. 1, pp.11-44, 1989.

[11] R.L. Birdwhistell: *Kinesics and Context: Essays on Body Motion Communication*, University of Pennsylvania Press, 1970.

[12] A. Cipriani e M. Giri: *Musica elettronica e sound design*, Vol. 2, pp. 435–507, ConTempoNet, 2013.

[6] Esempi di applicazioni audio e download dell'external Java: http://www.renatomessina.it/?p=328.