PIECEWISE DERIVATIVE ESTIMATION OF TIME-VARYING SINUSOIDS AS SPLINE EXPONENTIAL FUNCTIONS

Wen Xue
Samsung Electronics

xue.wen@samsung.com

ABSTRACT

This paper discusses the estimation of non-stationary sinusoidal parameters. We formulate a piecewise version of the distributive derivative algorithm, which is used to analyse non-stationary sinusoidal signals and estimate their frequencies and log amplitude derivatives over a long duration as spline functions, and apply this algorithm for the estimation of instantaneous frequencies, amplitudes and phase angles. Test results show that the piecewise derivative algorithm provides better estimation than the previous non-piecewise version at lower computation cost.

1. INTRODUCTION

The sinusoidal modelling technique [1][2] uses slow-varying sinusoids to model the "deterministic" parts of audio and speech, including harmonic partials of human/animal vocals, string, wind and brass instruments, and harmonic or inharmonic partials of percussion and electric instruments.

Complex exponential functions, or *complex sinusoids*, are of the form $e^{r(t)}$, where $r(t)=p(t)+j\varphi(t)$, $p(t)$, $\varphi(t)\in\mathbf{C}^1(\mathbf{R})$ [1], is the *exponent*. $e^{p(t)}$ and $\varphi(t)$ are known as the amplitude and phase angle, and $\omega(t)=d\varphi(t)/dt$ the angular frequency. We say $e^{r(t)}$ is slow-varying if $p(t)$ and $\omega(t)$ vary slowly with time t. Slow-varying sinusoids have narrow short-time bandwidths [3], allowing concurrent sinusoids be accessed independently via adequate band-pass mechanism, as long as their frequencies stand apart. In particular, the real sinusoid $e^{p(t)}\cos\varphi(t)$ can be accessed via $e^{r(t)}$.

Many sinusoid estimators derived in the past estimate sinusoidal parameters at a point from waveform data in its close vicinity. Several early estimators assuming short-term stationarity of amplitude and frequency were summarized in [4]. As real-world sinusoids are rarely stationary, more complex short-term parametric models, e.g. [5]-[9], were proposed, leading to the highly flexible estimation of arbitrary complex polynomial exponential functions [8][9], and more recently to an even higher degree of freedom by allowing an arbitrary complex polynomial multiplier on top of it [10].

A second family of algorithms addresses long-term amplitude and frequency modulations, e.g. with the spline model [11][12]. While most short-term algorithms engage closed-form computation, the long-term methods depend on iterative optimization, and are likely to suffer high computation cost and convergence to local optima. However, upon successful convergence the long-term constraints help to fight overfitting and improve robustness.

In this paper we show that the distributive derivative approach of [9] can be formulated to address long-term amplitude and frequency modulations using a spline exponential model. This leads to a non-iterative algorithm for the long-term estimation of sinusoids, which combines the simplicity of the derivative method with the robustness of long-term parametric modelling.

The rest of this paper is arranged as follows. Section 2 briefly reviews the distributive derivative algorithm; section 3 derives the piecewise formulation of the derivative algorithm, which estimates frequency and log amplitude derivative as splines; section 4 presents an amplitude and phase handling scheme that helps make long-term and local estimates consistent; section 5 presents test results on a synthesized test set, and section 6 presents a real world example.

[1] $\mathbf{C}^m(\mathbf{R})$: space of real functions with continuous mth-order derivatives.

2. THE DISTRIBUTIVE DERIVATIVE METHOD

The distributive derivative algorithm, or derivative algorithm for short, estimates a time-varying exponent $r(t)$ of $s(t)=e^{r(t)}$ by taking the derivative of $s(t)$. Early examples of the method include [13] and [14] used for estimating stationary sinusoids and linear chirps. Later the method was generalized by current author and Sandler [8] and Betser [9] to estimate $r(t)$ as the linear combination of differentiable functions. This section gives a brief review of this algorithm, following the formulation of [9].

2.1. General framework (after [9])

Let $h_1(t)$, …, $h_M(t)\in\mathbf{C}^1(\mathbf{R})$ be M linearly independent complex functions and $v_1(t)$, …, $v_I(t)\in\mathbf{C}^1(\mathbf{R})$ be I linearly independent complex functions, and let all these functions have a common compact support $D=[d_1,d_2]$, d_1, $d_2\in\mathbf{R}$. We consider a complex sinusoid

$$s(t)=e^{r(t)} \tag{1}$$

on the interval D, so that the derivative of $r(t)$ is a linear combination of $h_1(t),\ldots h_M(t)$:

$$r'(t)=\sum_{m=1}^{M}\lambda_m h_m(t)\equiv\mathbf{h}(t)^{\mathsf{T}}\boldsymbol{\lambda}\text{, } t\in D \tag{2}$$

where $\mathbf{h}(t)=(h_1(t),\ \ldots,\ h_M(t))^{\mathsf{T}}$, $\boldsymbol{\lambda}_M=(\lambda_1,\ \ldots,\ \lambda_M)^{\mathsf{T}}\in\mathbf{C}^M$ and the superscript $^{\mathsf{T}}$ denotes matrix and vector transpose. Now we consider this problem: given $s(t)$ and $\mathbf{h}(t)$, how do we find $\boldsymbol{\lambda}$?

We take the derivatives of both sides of (1) and substitute (2):

$$s'(t)=s(t)\cdot\mathbf{h}(t)^{\mathsf{T}}\boldsymbol{\lambda} \tag{3}$$

Taking the inner products of both sides of (3) with functions v_1, …, v_I we get

$$\left\langle s',v_i\right\rangle_{\mathrm{c}}=\left\langle s\mathbf{h}^{\mathsf{T}},v_i\right\rangle_{\mathrm{c}}\boldsymbol{\lambda}\text{, } i=1,\ \ldots,\ I \tag{4}$$

or

$$\left\langle s',\mathbf{v}\right\rangle_{\mathrm{c}}=\left\langle s\mathbf{h}^{\mathsf{T}},\mathbf{v}\right\rangle_{\mathrm{c}}\boldsymbol{\lambda}\text{,} \tag{5}$$

where $\mathbf{v}(t)=(v_1(t), \ldots, v_I(t))^\mathsf{T}$ and the continuous inner product operator $\langle \bullet,\bullet \rangle_c$ is defined for functions and function vectors as

$$\langle x,y\rangle_c = \int y^*(t)x(t)dt \text{ and } \langle \mathbf{x}^\mathsf{T},\mathbf{y}\rangle_c = \int \mathbf{y}^*(t)\mathbf{x}^\mathsf{T}(t)dt, \tag{6}$$

respectively, where the superscript * denotes complex conjugate. Comparing (3) and (5) we see that $\langle s',\mathbf{v}\rangle_c$ is the linear combination of vectors $\langle sh_m,\mathbf{v}\rangle_c$, $m=1, \ldots, M$, with the *same coefficients* as $r'(t)$ is that of $h_m(t)$. This converts the decomposition of $r'(t)$ in a function space $\{\, \mathbf{h}(t)^\mathsf{T}\boldsymbol{\lambda}\,|\boldsymbol{\lambda}\in\mathbf{C}^M\}$ to that of $\langle s',\mathbf{v}\rangle_c$ in a vector space without the need for extracting $r'(t)$ explicitly. We call the entries of $\mathbf{h}$ *basis functions* as they form a basis of the vector space above, and the entries of $\mathbf{v}$ *test functions* after [9].

2.2. Discrete computation

If we define discrete inner products for functions and function vectors as

$$\langle x,y\rangle = \sum_{n=0}^{T-1} y^*(n)x(n),\ \langle \mathbf{x}^\mathsf{T},\mathbf{y}\rangle = \sum_{n=0}^{T-1}\mathbf{y}^*(n)\mathbf{x}^\mathsf{T}(n), \tag{7}$$

then the following discrete version of (5) holds:

$$\langle s',\mathbf{v}\rangle = \langle s\mathbf{h}^\mathsf{T},\mathbf{v}\rangle\boldsymbol{\lambda} \tag{8}$$

One issue of using (8) to compute $\boldsymbol{\lambda}$ is that $s'(t)$ is not available as input. Both [8] and [9] suggested that if $\mathbf{v}(t)$ is differentiable and vanishes at both ends of D, then $\langle s',\mathbf{v}\rangle_c$ can be computed as $-\langle s,\mathbf{v}'\rangle_c$. For discrete computation however, $-\langle s,\mathbf{v}'\rangle$ approximates $\langle s',\mathbf{v}\rangle$ with an error given in [9] as

$$-\langle s,\mathbf{v}'\rangle = \langle s',\mathbf{v}\rangle - \sum_n (s\mathbf{v}^*)'(t)\,|_{t=n} \tag{9}$$

In [8] we pointed out that this error equals the total (Shannon) sampling alias of $(s\mathbf{v}^*)(t)$. To keep the error term in (9) low, $(s\mathbf{v}^*)(t)$ must have negligible spectral energy density above the Nyquist frequency. Practically this is satisfied by choosing $v_i\in\mathbf{C}^2(\mathbf{R})$, $\forall i$, so that $s(t)v_i(t)^*$ is a base-band signal: for example, a Hann-windowed sinusoid tuned to the central frequency of $s(t)$.

In practice the coefficients of (8) may be contaminated by noise in $s(t)$ (observation noise) and $r'(t)$ (modelling noise). As a remedy it is often solved in a least-square sense using

$$\boldsymbol{\lambda} = \left(\langle \mathbf{v}^\mathsf{T},s\mathbf{h}\rangle\langle s\mathbf{h}^\mathsf{T},\mathbf{v}\rangle\right)^{-1}\langle \mathbf{v}^\mathsf{T},s\mathbf{h}\rangle\langle s',\mathbf{v}\rangle, \tag{10}$$

provided that $\langle \mathbf{v}^\mathsf{T},s\mathbf{h}\rangle\langle s\mathbf{h}^\mathsf{T},\mathbf{v}\rangle$ is invertible.

The discrete inner products can also be written as matrix multiplications. Define $\mathbf{t}=(0,\ldots,T\text{-}1)^\mathsf{T}$, $\mathbf{s}=s(\mathbf{t})=(s(0),\ldots,s(T\text{-}1))^\mathsf{T}$, $\mathbf{s}'=s'(\mathbf{t})$, $\mathbf{H}=\mathbf{h}(\mathbf{t}^\mathsf{T})=(\mathbf{h}(0),\ldots,\mathbf{h}(T\text{-}1))$, $\mathbf{V}=\mathbf{v}(\mathbf{t}^\mathsf{T})$, then (8) has the matrix formulation

$$\mathbf{V}^*\mathbf{s}'=\mathbf{V}^*diag(\mathbf{s})\mathbf{H}^\mathsf{T}\boldsymbol{\lambda}. \tag{11}$$

2.3. Amplitude and phase angle

The derivative algorithm only estimates $r'(t)$. To complete the estimation of $s(t)=e^{r(t)}$ we still need to estimate $r(0)$, which represents global amplification and phase shift. The least-square estimate of $r(0)$ is computed by correlation with a unit-amplitude zero-phase sinusoid with exponent derivative $r'(t)$:

$$r(0)=\log\frac{\langle s,\tilde{s}\rangle}{\langle \tilde{s},\tilde{s}\rangle},\ \tilde{s}(t)=\exp\int_0^t \mathbf{h}(\tau)^\mathsf{T}\boldsymbol{\lambda}d\tau \tag{12}$$

The instantaneous angular frequency, amplitude and phase angle at 0 are $\mathrm{Im}\{r'(0)\}$, $e^{\mathrm{Re}\{r(0)\}}$ and $\mathrm{Im}\{r(0)\}$, respectively.

3. PIECEWISE DERIVATIVE METHOD

The derivative algorithm above assumes that $r'(t)$ follows the same parametric model over the whole duration. [9] took $\mathbf{h}(t)$ as a polynomial basis, i.e. $\mathbf{h}(t)=(t^{M\text{-}1},t^{M\text{-}2},\ldots,1)^\mathsf{T}$, and asserted the validity of the signal model (2) by the Taylor expansion of $r'(t)$. Since Taylor expansions are usually accurate only in the neighbourhood of the origin, this model is not suitable for long duration. On the other hand, piecewise polynomials, or splines, can model arbitrary functions of arbitrary lengths using translations of the same local model. In this section we adapt the derivative algorithm to estimate $r'(t)$ as a spline function. To distinguish between the original and adapted versions, we call them *local* and *piecewise* derivative algorithms, respectively.

3.1. General framework

We limit our discussion to splines with uniformly placed *knot points*, and let them be 0, T, …, LT. We assume that we know the waveform of a slow-varying sinusoid on $[0,LT]$ and roughly know its instantaneous frequencies at 0, T, … LT, both of which can be obtained using a sinusoid tracker, e.g. [1].

We formulate $r'(t)$ as a spline function by expressing it as a $(M\text{-}1)$th-order polynomial on each $[lT,lT+T)$, $l=0, \ldots, L\text{-}1$:

$$r'(lT+t)=\mathbf{h}(t)^\mathsf{T}\boldsymbol{\lambda}_l,\ 0\le t<T,\ l=0, \ldots, L\text{-}1, \tag{13}$$

where $\mathbf{h}(t)=(t^{M\text{-}1},t^{M\text{-}2},\ldots,1)^\mathsf{T}$, $\lambda_l\in\mathbf{C}^M$. Vectors $\boldsymbol{\lambda}_0$, ..., $\boldsymbol{\lambda}_{L\text{-}1}$ contain the polynomial coefficients on the L segments, which are constrained by boundary conditions specific to the spline type. For example, the continuity of $r'(t)$ at lT requires

$$\mathbf{h}(T)^\mathsf{T}\boldsymbol{\lambda}_{l-1}=\mathbf{h}(0)^\mathsf{T}\boldsymbol{\lambda}_l,\ l=1, \ldots, L\text{-}1. \tag{14}$$

In this paper we consider the linear interpolative formulation[2] of splines, which expresses $\boldsymbol{\lambda}_l$ as a linear function of the spline samples at the knot points, i.e. $\mathbf{r}'=r'(\mathbf{t})=(r'(0),\ldots, r'(LT))^\mathsf{T}$:

$$\boldsymbol{\lambda}_l=\mathbf{A}_l\mathbf{r}',\ l=0, \ldots, L\text{-}1. \tag{15}$$

$\mathbf{A}_0$, …, $\mathbf{A}_{L\text{-}1}$ are real matrices depending on L, T and the spline type. Substituting (15) into (13) we get

$$r'(lT+t)=\mathbf{h}(t)^\mathsf{T}\mathbf{A}_l\mathbf{r}',\ 0\le t<T,\ l=0, 1, \ldots, L\text{-}1. \tag{16}$$

We call (13)~(16) the spline exponential model. A specific spline interpolator has a linear interpolative formulation (15) as long as all its boundary conditions are linear in terms of $r'(t)$ and its derivatives. The matrices $\mathbf{A}_0$, …, $\mathbf{A}_{L\text{-}1}$ for linear, quadratic and cubic spline interpolators are derived in the Appendix.

We further define a function vector $\boldsymbol{\rho}(t)=(\rho_0(t),\ldots, \rho_L(t))^\mathsf{T}$ by

$$\boldsymbol{\rho}(lT+t)^\mathsf{T}=\mathbf{h}(t)^\mathsf{T}\mathbf{A}_l,\ 0\le t<T,\ l=0, 1, \ldots, L\text{-}1. \tag{17}$$

If both $\mathbf{h}(t)$ and $\mathbf{A}_l$ are known then so is $\boldsymbol{\rho}(t)$. The independent coefficients $\mathbf{r}'$ contribute to $r'(t)$ through the entries of $\boldsymbol{\rho}(t)$:

$$r'(t)=\boldsymbol{\rho}(t)^\mathsf{T}\mathbf{r}',\ 0\le t<LT. \tag{18}$$

Eq. (18) expresses the linear piecewise exponential model as a special case of the linear exponential model (2), with $\boldsymbol{\rho}(t)$ replacing $\mathbf{h}(t)$ as the basis. We can therefore construct a linear system similar to (8)

$$\langle s',\mathbf{v}\rangle = \langle s\boldsymbol{\rho}^\mathsf{T},\mathbf{v}\rangle\mathbf{r}'. \tag{19}$$

where $\mathbf{v}(t)=(v_1(t), \ldots, v_{\dim(\mathbf{v})}(t))^\mathsf{T}$ is the vector of test functions. If $\langle \mathbf{v}^\mathsf{T},s\boldsymbol{\rho}\rangle\langle s\boldsymbol{\rho}^\mathsf{T},\mathbf{v}\rangle$ is invertible, then (19) has a least square solution

[2] B-spline formulation is also a possibility. The interpolative expression is chosen because it preserves the task's degree of freedom and allows us draw links with previous methods, like (18).

$$\mathbf{r'} = \left(\left\langle \mathbf{v}^{\mathsf{T}}, s\boldsymbol{\rho}\right\rangle \left\langle s\boldsymbol{\rho}^{\mathsf{T}}, \mathbf{v}\right\rangle\right)^{-1} \left\langle \mathbf{v}^{\mathsf{T}}, s\boldsymbol{\rho}\right\rangle \left\langle s', \mathbf{v}\right\rangle . \tag{20}$$

Equations (19) and (20) give the *piecewise derivative algorithm* for estimating sinusoids with model (16). Notice that although we focus on spline exponentials in this paper, the algorithm itself does not require $\mathbf{h}(t)$ to be a polynomial basis, and therefore can be applied to a larger class of piecewise models of $r'(t)$, as long as they can be formulated as (16).

3.2. Computing coefficient matrix $\langle s\boldsymbol{\rho}^{\mathsf{T}},\mathbf{v}\rangle$

While one can always compute $\boldsymbol{\rho}(t)$ explicitly with (17), the actual estimation of $\mathbf{r'}$ only requires computing the matrix $\langle s\boldsymbol{\rho}^{\mathsf{T}},\mathbf{v}\rangle$, which has a piecewise implementation:

$$\begin{aligned}\left\langle s\boldsymbol{\rho}^{\mathsf{T}}, \mathbf{v}\right\rangle &= \sum_{t=0}^{LT-1} \mathbf{v}^{*}(t)s(t)\boldsymbol{\rho}(t)^{\mathsf{T}} = \sum_{l=0}^{L-1}\sum_{t=0}^{T-1} \mathbf{v}^{*}(lT+t)s(lT+t)\boldsymbol{\rho}(lT+t)^{\mathsf{T}} \\ &= \sum_{l=0}^{L-1}\sum_{t=0}^{T-1} \mathbf{v}^{*}_{[l]}(t)s_{[l]}(t)\mathbf{h}(t)^{\mathsf{T}}\mathbf{A}_l = \sum_{l=0}^{L-1}\left\langle s_{[l]}\mathbf{h}^{\mathsf{T}}, \mathbf{v}_{[l]}\right\rangle \mathbf{A}_l\end{aligned} \tag{21}$$

where $\mathbf{v}_{[l]}(t)=\mathbf{v}(lT+t)$ and $s_{[l]}(t)=s(lT+t)$ represent the parts of $\mathbf{v}(t)$ and $s(t)$ sampled over interval $[lT,lT+T)$.

3.3. Separate models for amplitude and frequency

We let $\mathbf{h}(t)$ be real and replace (15) with

$$\boldsymbol{\lambda}_l = \mathbf{B}_l\mathbf{p'} + j\mathbf{C}_l\boldsymbol{\omega}, \quad l=0, \dots, L\text{-}1, \tag{22}$$

where $\mathbf{p'}=\mathrm{Re}\{\mathbf{r'}\}$ and $\boldsymbol{\omega}=\mathrm{Im}\{\mathbf{r'}\}$ contain the amplitude growth rates and angular frequencies at 0, T, ..., LT, respectively, and $\mathbf{B}_l$ and $\mathbf{C}_l$, l=0, ..., L-1, are real matrices that implement linear interpolations of $\mathbf{p'}$ and $\boldsymbol{\omega}$ via (13). This formulation allows the amplitude and frequency be modelled with independent spline types, and leads to the real implementation of the piecewise derivative method:

$$\begin{pmatrix} \mathrm{Re}\left\langle s\boldsymbol{\rho}_{\mathbf{B}}^{\mathsf{T}}, \mathbf{v}\right\rangle & -\mathrm{Im}\left\langle s\boldsymbol{\rho}_{\mathbf{C}}^{\mathsf{T}}, \mathbf{v}\right\rangle \\ \mathrm{Im}\left\langle s\boldsymbol{\rho}_{\mathbf{B}}^{\mathsf{T}}, \mathbf{v}\right\rangle & \mathrm{Re}\left\langle s\boldsymbol{\rho}_{\mathbf{C}}^{\mathsf{T}}, \mathbf{v}\right\rangle \end{pmatrix} \begin{pmatrix}\mathbf{p'} \\ \boldsymbol{\omega}\end{pmatrix} = \begin{pmatrix}\mathrm{Re}\left\langle s', \mathbf{v}\right\rangle \\ \mathrm{Im}\left\langle s', \mathbf{v}\right\rangle\end{pmatrix}, \tag{23}$$

where

$$\boldsymbol{\rho}_{\mathbf{B}}(lT+t)^{\mathsf{T}}=\mathbf{h}(t)^{\mathsf{T}}\mathbf{B}_l,\ \boldsymbol{\rho}_{\mathbf{C}}(lT+t)^{\mathsf{T}}=\mathbf{h}(t)^{\mathsf{T}}\mathbf{C}_l,\ 0\le t<T,\ l=0, \dots, L\text{-}1. \tag{24}$$

A least square solution to (23) is computed in the same way as to (8) using real arithmetic only. $\langle s\boldsymbol{\rho}_{\mathbf{B}}^{\mathsf{T}},\mathbf{v}\rangle$ and $\langle s\boldsymbol{\rho}_{\mathbf{C}}^{\mathsf{T}},\mathbf{v}\rangle$ are computed using (21) from the same set of intermediate results $\langle s_{[l]}\mathbf{h}^{\mathsf{T}},\mathbf{v}_{[l]}\rangle$, l=0, …, L-1.

3.4. Framing of test functions v(t)

In this section we present a specific construction of the test functions $\mathbf{v}(t)$ using overlapping frames. We wish the interval $[0,LT]$ be uniformly covered by $\mathbf{v}(t)$, so that no part of $s(t)$ is ignored or overemphasized. It is intuitive to divide this interval into uniformly spaced frames and apply the same subset of test functions to every frame. And as test functions must vanish at both ends, it is necessary to have overlapping frames. In this paper we place frame centres at the spline knots, i.e. T, $2T$, …, $(L\text{-}1)T$, with 50% overlap between adjacent frames. This gives L-1 frames of length $2T$ over the whole duration (Fig.1a). Given this framing scheme we can write $\mathbf{v}(t)^{\mathsf{T}}=[\mathbf{v}_1(t)^{\mathsf{T}}\ \mathbf{v}_2(t)^{\mathsf{T}} \dots \mathbf{v}_{L\text{-}1}(t)^{\mathsf{T}}]$, in which all entries of $\mathbf{v}_l(t)$, l=1, …, L-1, are supported on $[lT\text{-}T,lT+T]$, and are time-shifted versions of the same *local* test functions:

$$\mathbf{v}_l(lT+t)= \mathbf{v}_1(T+t),\ \ l=1, ..., L\text{-}1,\ \ \text{-}T\le t\le T. \tag{25}$$

Eq.(25) reduces the design of $\mathbf{v}(t)$ to that of $\mathbf{v}_1(t)$, for which the test functions in the local derivative method (section 2), e.g. windowed Fourier atoms, can be used unchanged.

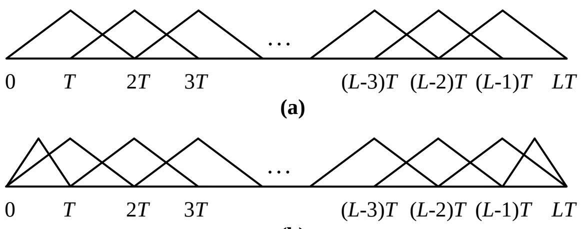


Figure 1 . Framing without and with boundary frames
(a) without boundary frames; (b)with boundary frames

As Figure1(a) shows, the vanishing requirement of test functions inevitably leads to poor frame coverage near 0 and LT. To make better use of data in these parts we add two *boundary frames* supported on $[0,T]$ and $[LT\text{-}T,LT]$, respectively, and construct test functions on these frames as

$$\mathbf{v}_0(t)= \mathbf{v}_L(LT\text{-}T+t)=\mathbf{v}_1(2t), \tag{26}$$

as illustrated in Figure1(b).

3.5. Computational complexity

We examine the number of complex multiplications in the general formulation of the piecewise derivative algorithm, finishing with solving (20) using boundary frames as shown in Figure1(b). Let I be the number of test functions per frame. The computation considered includes computing the coefficients of (19), and solving (19) using (20).

Using boundary frames, $2I$ test functions are non-zero on each of the L sections of size T. Accordingly, $2TI(M+1)$ multiplications are needed to compute $\langle s_{[l]}\mathbf{h}^{\mathsf{T}},\mathbf{v}_{[l]}\rangle$ for each l. $L(L+1)IM$ more multiplications are needed to compute $\langle s\boldsymbol{\rho}^{\mathsf{T}},\mathbf{v}\rangle$ using (21). Computing $\langle s',\mathbf{v}\rangle$ requires $2LTI$ multiplications, i.e. the total length of all test functions. The total number of multiplications for computing (19) sums up to $2LT(M+2)I+L(L+1)MI$. Computing $\langle \mathbf{v}^{\mathsf{T}},s\boldsymbol{\rho}\rangle\langle s',\mathbf{v}\rangle$ and $\langle \mathbf{v}^{\mathsf{T}},s\boldsymbol{\rho}\rangle\langle s\boldsymbol{\rho}^{\mathsf{T}},\mathbf{v}\rangle$ require LNI and $L(L+1)^2I/2$ multiplications respectively; solving (20) by Gaussian elimination requires $L(L+1)(2L+7)/6$.

In comparison, if we apply the local derivative algorithm over $[0,LT]$ with boundary frames, i.e. applying the algorithm to each of the L-1 frames of size $2T$ plus 2 frames of size T, then the coefficients of (8) are computed L+1 times for a total signal duration of $2LT$, and the linear system (8) is solved L+1 times.

Table 1. Complexity of piecewise and local derivative algorithms

	piecewise derivative alg.	local derivative alg.
computing coefficients	$2LTI(M+2)+L(L+1)MI$	$2LTI(M+2)$
solving linear system(s)	$L^3(I/2+1/3)$	$LM^2(I+M/3)$

Table 1 compares the complexity of the two derivative methods, in which we have ignored less significant terms. In typical applications M and I are small numbers (usually below 10), T is a few orders of magnitude larger, while L has a flexible range. As long as L is not too large, the complexity of both algorithms are dominated by $2LTI(M+2)$. In this case the piecewise derivative algorithm saves computation by allowing I be as small as 1, while $I\ge M$ must be satisfied in the local derivative algorithm. This benefit will be lost when L grows near $M\sqrt{2T}$, as the extra computation spent on solving (20) eventually outgrows the saving.

4. AMPLITUDE AND PHASE

Like its local counterpart, the piecewise derivative algorithm only estimates $r'(t)$. An alternative algorithm, such as the correlation method (12) in 2.3, must be employed to determine the global amplification and phase shift.

While in theory estimating $r(t)$ at any point, e.g. $r(0)$, is enough for reconstructing $r(t)$ by integrating $r'(t)$, doing so accumulates potential errors over time. As piecewise models are designed for long signals, the accumulated error can get dramatic. For this reason we propose to estimate $r(t)$ locally from short intervals at various measurement points, then adjust the $r'(t)$ estimates to fit the local $r(t)$ estimates. We present the details using the separated formulation (22).

4.1. Local estimation

We estimate $r(t)$ at knots lT, l=0, …, L, by applying the correlation method (12) to a short interval near lT:

$$r_l = \tilde{r}(lT) + \log\frac{\langle s, w_l\tilde{s}\rangle}{\langle \tilde{s}, w_l\tilde{s}\rangle},\ \ l=0,\ \dots,\ L, \tag{27}$$

where r_l is the local estimate of $r(lT)$, w_l is a window function on $[lT\text{-}T,lT\text{+}T]$ for l=1, .., L-1, on $[0,T]$ for l=0, and on $[LT\text{-}T,LT]$ for l=L, and

$$\tilde{r}(t) = \int_0^t \mathbf{h}(t)^{\mathsf{T}}(\mathbf{B}_l\mathbf{p}'+j\mathbf{C}_l\boldsymbol{\omega})d\tau,\ \ \tilde{s}(t) = \exp\tilde{r}(t). \tag{28}$$

While these estimates can be used as they are, in applications like additive synthesis it is desirable that the local estimates of $r(t)$ be coherent with the piecewise estimate of $r'(t)$, i.e. $\exists k_l\in\mathbf{Z}$, l=0, …, L-1, so that

$$\int_{lT}^{lT+T} r'(t)dt = \int_0^T \mathbf{h}(t)^{\mathsf{T}}(\mathbf{B}_l\mathbf{p}'+j\mathbf{C}_l\boldsymbol{\omega})dt = r_{l+1} - r_l + j2k_l\pi\ . \tag{29}$$

This is achievable using an adjustment step described below, which applies a fine tuning to $(\mathbf{p}',\boldsymbol{\omega})$ to satisfy (29).

4.2. Adjustment of ω and p'

Let $\boldsymbol{\omega}$ be the original frequency estimate and $\boldsymbol{\omega}+\boldsymbol{\psi}$ be the adjusted estimate. Define

$$\mathbf{u}_l = \int_0^T \mathbf{C}_l^{\mathsf{T}}\mathbf{h}(t)dt\ ,\ \ l=0,\ \dots,\ L\text{-}1. \tag{30}$$

Taking the imaginary part of both sides of (29) we get

$$\mathbf{u}_l^{\mathsf{T}}(\boldsymbol{\omega}+\boldsymbol{\psi}) = \mathrm{Im}\{r_{l+1}-r_l\} + 2k_l\pi\ ,\ \ l=0,\ \dots,\ L\text{-}1. \tag{31}$$

We start with a phase unwrapping process similar to that of [1] to implicitly determine k_l in (31), which we rewrite as

$$\mathbf{u}_l^{\mathsf{T}}\boldsymbol{\psi} = \mathrm{Im}\{r_{l+1}-r_l\} - \mathbf{u}_l^{\mathsf{T}}\boldsymbol{\omega} + 2k_l\pi\ ,\ \ l=0,\ \dots,\ L\text{-}1. \tag{32}$$

$\mathbf{u}_l^{\mathsf{T}}\boldsymbol{\psi}$ is the integral of an interpolation of $\boldsymbol{\psi}$ over $[lT,lT+T]$. As $\boldsymbol{\psi}$ represents a fine adjustment we choose the $k_l\in\mathbf{Z}$ that minimizes the right side of (32), which becomes

$$\mathbf{u}_l^{\mathsf{T}}\boldsymbol{\psi} = res(\mathrm{Im}\{r_{l+1}-r_l\} - \mathbf{u}_l^{\mathsf{T}}\boldsymbol{\omega}, 2\pi)\ ,\ \ l=0,\ \dots,\ L\text{-}1, \tag{33}$$

where $res(x,2\pi)$ is the minimal-absolute residue of x modular 2π. Notice that this choice of k_l coincides with that of [1] motivated by maximizing phase smoothness.

Define $b_l = res(\mathrm{Im}\{r_{l+1}-r_l\} - \mathbf{u}_l^{\mathsf{T}}\boldsymbol{\omega}, 2\pi)$, $\mathbf{b}=(b_0,\ \dots,b_{L\text{-}1})^{\mathsf{T}}$, and $\mathbf{U}=(\mathbf{u}_0,\dots,\mathbf{u}_{L\text{-}1})$, then (33) is simplified to

$$\mathbf{U}^{\mathsf{T}}\boldsymbol{\psi} = \mathbf{b}. \tag{34}$$

This is a linear system of $\boldsymbol{\psi}$ with L equations and L+1 unknown variables. Since $\boldsymbol{\psi}$ is expected to be small, a simple choice is the minimal-norm solution, given by

$$\boldsymbol{\psi} = \mathbf{U}(\mathbf{U}^{\mathsf{T}}\mathbf{U})^{-1}\mathbf{b}. \tag{35}$$

Finally the adjustment of $\boldsymbol{\omega}$ is completed by

$$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} + \boldsymbol{\psi}\ . \tag{36}$$

It is trivial to verify the adjusted $\boldsymbol{\omega}$ satisfies (31).

Adjustment of $\mathbf{p}'$ follows the same procedure as above, except that the phase unwrapping step is not needed.

5. EXPERIMENTS

We test the proposed algorithm on synthesized test signals and compare it to the local derivative algorithm [9], the QUASAR estimator [11], and original quadratic-interpolated fast Fourier transform magnitude (QIFFT) method [15]. The piecewise derivative algorithm is tested with cubic and linear splines (labelled PD^3 and PD^1), the local derivative algorithm is tested with cubic and linear polynomials (LD^3 and LD^1), while the QUASAR estimator (Q) is piecewise linear by design (as "quadratic phase" means linear frequency).

We use each estimator to estimate parameters of a slow-varying sinusoid, then reconstruct the sinusoid from the estimates with a paired synthesizer. Errors are computed for estimated parameters and for sinusoids synthesized from them. The local estimators LD^3, LD^1 and QIFFT (QIF) do not come with “native” synthesizers for more than one frame. We pair LD^3 and LD^1 with natural cubic and linear spline interpolators respectively, and QIFFT with the original McAulay-Quatieri phase-aligned synthesizer [1]. For the reconstruct errors we also test a few *reference* systems that synthesize from the true parameters. Reference system using cubic and linear spline synthesizers are labelled R^3 and R^1 respectively. [11] provides its own least-square interpolator, which we also use as a reference ("R^{Q}").

Both derivative algorithms use boundary frames. From 3.4, the standard frame size is $2T$; standard frame hop and boundary frame size are T. Two test functions per frame are used in the piecewise derivative algorithms (PD^3, PD^1); four per frame are used in the local derivative algorithms (LD^3, LD^1).

For each test signal we use its waveform and reference frequencies at measurement points 0, T, …, LT as inputs. Test signals over $[0,LT]$ are supplied to the derivative algorithms; test signals over $[\text{-}T,LT\text{+}T]$ are supplied to the Q and QIF estimators which do not fully handle boundary frames. Reference frequencies are rough frequency estimates to tell the estimators where in the T-F plane to expect the sinusoids. We provide these by rounding ground truth frequencies to nearest whole DFT bins, 1bin=$1/2T$. The QUASAR estimator (Q) uses a single reference frequency, which we supply with the average ground value.

For test functions of both derivative algorithms we follow [9] and use Hann-windowed complex sinusoids whose frequencies are tuned to the whole DFT bins closest to the reference frequencies.

5.1. Test set

Frequency and amplitude-modulated sinusoids are used as test signals. The frequencies are modulated by a sinusoidal modulator of amplitude A_{M} and period T_{M}; the amplitudes are modulated by passing the modulated frequency through one of three real transfer functions: a linear function $H_1(f)$ that assigns the central frequency a medium amplitude, and two quadratic functions $H_2(f)$ and $H_3(f)$ that assign the central frequency either the minimal or maximal amplitude. We summarize the test set by (37a)~(37f), where f, f_0, A_M are given in bins, and T_M in frames, where 1 bin=1/(2T), 1 frame=T. Constant coefficients in (37d)~(37f) are chosen so that the amplitude modulation depth is 2/3. For all tests we use L=10 and T=1024, so that the length of each test sample is 10240. For this choice of L and T the complexity of both derivative algorithms is dominated by computing $\langle s\boldsymbol{\rho}^{T},\mathbf{v}\rangle$, so that by using I=2 instead of 4 the piecewise algorithm saves nearly half the computation.

Table 2. Synthesized test set

$$f(t) = f_0 + A_M \cos(\varphi_M + 2\pi t / TT_M)\,, \tag{37 a}$$

$$\varphi(t) = \frac{\pi}{T} f_0 t + \frac{T_M A_M}{2} \sin(\varphi_M + 2\pi t / TT_M)\,, \tag{37b}$$

$$a(t) = H_i(f(t))\,,\ \ i\text{=1, 2, 3;} \tag{37c}$$

$$H_1(f) = (f - f_0 + 1.5A_M) / A_M\,, \tag{37d}$$

$$H_2(f) = 0.5 + 2(f - f_0)^2 / A_M^2\,, \tag{37e}$$

$$H_3(f) = 2.5 - 2(f - f_0)^2 / A_M^2\,. \tag{37f}$$

For the tests we sample T_M uniformly at 6 positions between 5 and 15 frames, A_M logarithmically at 6 positions between 1 and 32 bins, φ_M uniformly at 6 positions between 0 and $5\pi/6$, f_0 uniformly at 10 positions between 155 and 155.9 bins. This makes a total of 6480 test signals. Apart from clean sinusoids, we also run tests on sinusoids contaminated by concurrent sinusoids or white noise. Reference systems are not tested with contaminated signals.

5.2. Results

We present the test results as accuracy measurements against selected parameters (T_M, A_M, estimator) averaged over all relevant test results. The measurements are amplitude accuracy (E_a), frequency accuracy (E_f) and signal-to-error ratio (*SER*). E_a is computed from the ground truth log amplitude derivatives p_0, …, p_L and their estimates $\tilde{p}_0$, …, $\tilde{p}_L$ as

$$E_a = -10\log_{10} \frac{1}{L+1} \sum_{l=0}^{L} (p_l - \tilde{p}_l)^2\,. \tag{38a}$$

E_f is computed from the ground truth angular frequencies ω_0, …, ω_L and their estimates $\tilde{\omega}_0$, …, $\tilde{\omega}_L$ as

$$E_f = -10\log_{10} \frac{1}{4\pi^2(L+1)} \sum_{l=0}^{L} (\omega_l - \tilde{\omega}_l)^2\,. \tag{38b}$$

SER is computed from the test signal $s(t)$ and resynthesized sinusoid $\tilde{s}(t)$ as

$$SER(dB) = -10\log_{10} \frac{\sum (s(t) - \tilde{s}(t))^2}{\sum \tilde{s}(t)^2}\,. \tag{38c}$$

Higher values of E_a, E_f and *SER* indicate better amplitude, frequency and reconstruction accuracy, respectively.

5.2.1. *Clean sinusoids*

Figure 2 compares results of tested estimators on clean signals. Each curve presents the result from one estimator, as *SER*, E_f or E_a against FM period T_M in (a)(c)(e), and against FM extent A_M in (b)(d)(f), averaged over all φ_M, f_0 and transfer functions $H_1(f)$~$H_3(f)$ used in (37c). *SER* results of reference systems are included in (a) and (b).

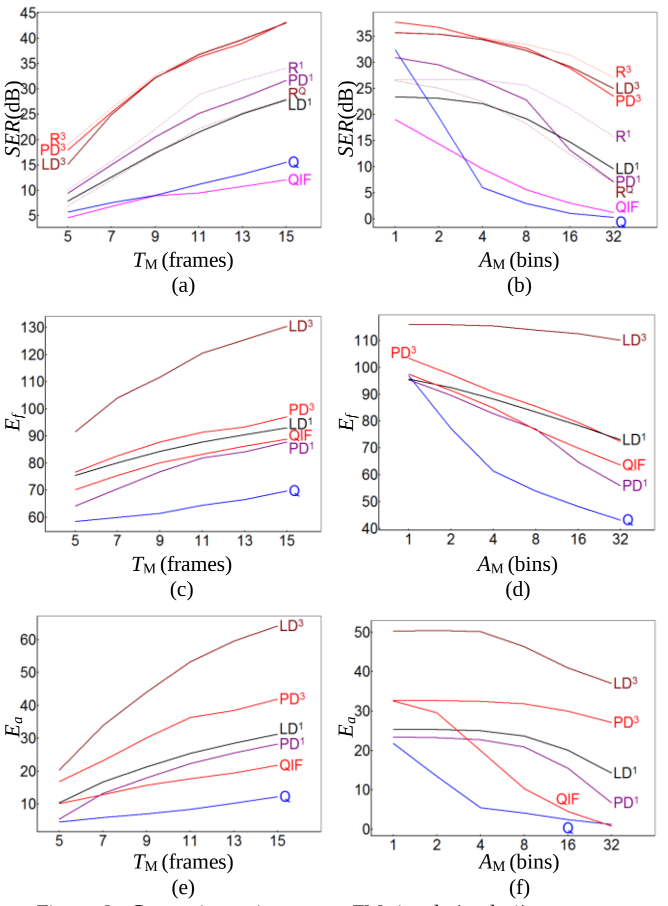


Figure 2 . Comparing estimators on FM signals (cycle 1)

As may be expected, derivative estimators using cubic polynomials outperform those using linear polynomials. Among estimators using the same polynomial orders, the local derivative algorithm does best in estimating log amplitude and frequency. This can be attributed to that fact that it uses more parameters to model the sinusoid (LD^1 uses 5 per frame, LD^3 uses 9, QIFFT use 3, all others use less than 2.1). However, this advantage is not observed in the *SER* results, which treat the whole duration equally instead of focusing on the measurement points. Since the test signals are off-model, better parameter estimation alone does not guarantee better modelling accuracy. This is even more distinctively observed in Fig. 2(b), which shows that the reference systems, in spite of holding the ground truth at measurement points, do not always have the highest *SER*. On the whole the two derivative algorithms provide similar *SER*s, and both PD^3 and LD^3 come close to R^3. Good results are observed from QUASAR estimator (Q) only if both A_M and $1/T_M$ are low. This can be traced back to the heterodyne filtering technique it uses to obtain its preliminary estimates, which cannot handle large frequency modulations.

5.2.2. *Test in the presence of other sinusoids*

In the presence of concurrent sinusoids, the main influence on the estimation comes from those with closest frequencies. In our experiments the test signal $s(t)$ is mixed with two concurrent sinusoids, with the same amplitude and initial phase as $s(t)$ and frequencies at ±B bins from $s(t)$. The frequency gap B is sampled logarithmically at 6 positions between 4 and 128 bins.

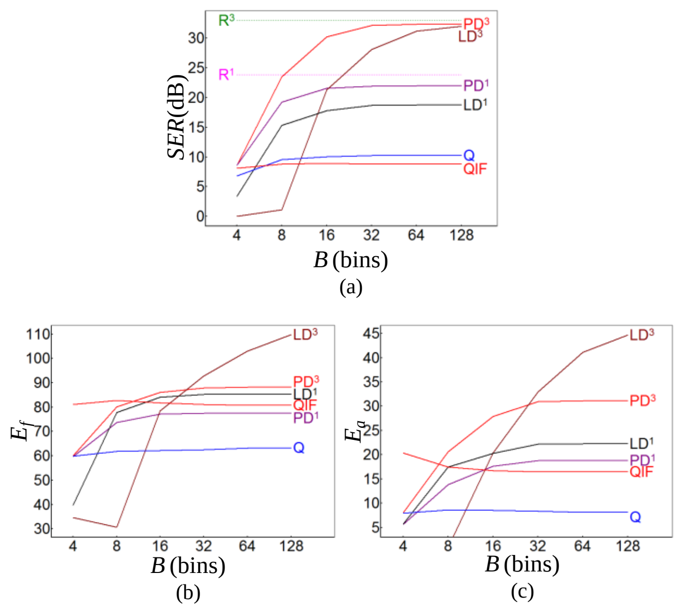


Figure 3 .Comparing estimators on FM signals with additional sinusoids

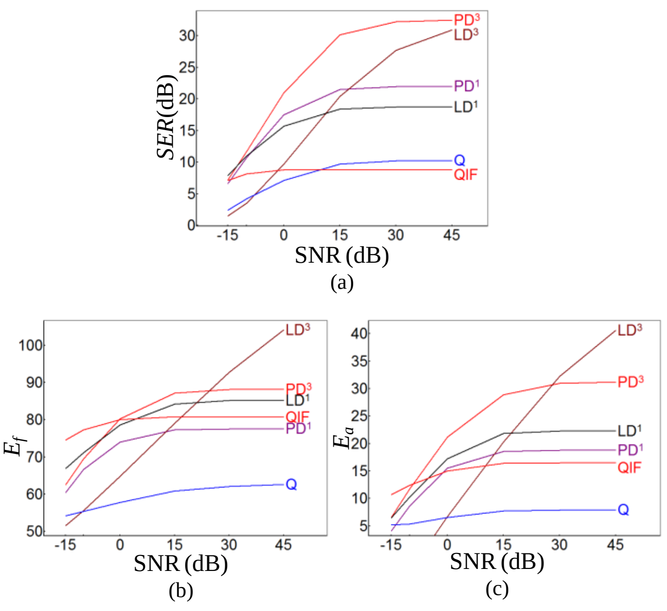


Figure 4 . Comparing estimators on FM signals with white noise

Figure 3 compares the results of all tested estimators, each curve presenting the result from one estimator, as *SER*, E_f or E_a against frequency gap B, averaged over all T_M, A_M, φ_M, f_0 and transfer functions $H_1(f)$~$H_3(f)$. While E_f and E_a results are averaged as they are, negative *SER*s are treated as 0 when computing the average. Results of direct synthesis R^3 and R^1 are included for reference. We see that concurrent sinusoids have significant impact on all tested estimators except QIF. For small values of B (i.e. strong interference) the piecewise derivative algorithm shows clear advantage in the presence of concurrent sinusoids: estimator PD^1 using a linear spline is already comparable to LD^3 using trinomials when $B \leq 16$. This can be attributed to the fact that the piecewise derivative algorithm uses less free parameters, and therefore is less likely to overfit. This explanation is also confirmed by LD^1 outperforming LD^3 under strong interference. The apparent resistance of QIF to disturbance is related to the construction of the test signals, as the choice of two symmetrical sinusoids on either side of $\omega(t)$ by whole DFT bins helps to minimize their total impact on the quadratic interpolation.

5.2.3. *Test in the presence of noise*

In this part we mix the test signals with Gaussian white noise at 6 levels, specified by 6 signal-to-noise ratios (SNRs) from -15dB to 45dB. Typical Cramer-Rao bounds for parameter estimation from noisy data are not included because our test signals are not synthesized to fit the parametric models.

Figure 4 compares the results of all tested estimators , each curve presenting the result from one estimator, as *SER*, E_f or E_a against input SNR, averaged over all T_M, A_M, φ_M, f_0 and transfer functions $H_1(f)$~$H_3(f)$. Negative *SER*s are treated as 0 when computing the average. Again, for small values of SNR (i.e. strong noise) the piecewise derivative algorithms shows consistent advantage over their local counterparts, and LD^1 once more outperforms LD^3. In the lower end of SNR the QIFFT method returns the best frequency estimate. This is the result of the estimate being explicitly bounded near the reference frequency no more than 0.5 bin from the ground truth, without much chance of large deviation even in extraordinary noise.

5.3. Summary

We have tested sinusoidal parameter estimation accuracy based on the piecewise derivative and local derivative algorithms, as well as the QUASAR estimator, using signals synthesized independent of the parametric models. For clean sinusoids the local derivative algorithm has the best parameter estimation accuracy, but the piecewise algorithm has the best reconstruction *SER*. The QUASAR estimator has similar performance to PD^1 only for very low modulation, and degrades quickly for high modulation. For sinusoids contaminated with noise or concurrent sinusoids, the piecewise derivative algorithm shows consistent improvement in estimation accuracy, which we attribute to long-term modelling and the reduction of the number of parameters. Notably, these improvements are achieved with half the computation cost of the local derivative method, making the piecewise version even more favourable.

Although we have treated the non-iterative and iterative estimators as mutually exclusive in the experiments, in reality they are not. A good non-iterative method, such as the piecewise derivative algorithm, can help provide an iterative algorithm an initial estimate close to global optimum, therefore relieves the latter of such potential disadvantages as heavy computation and convergence at local optimum.

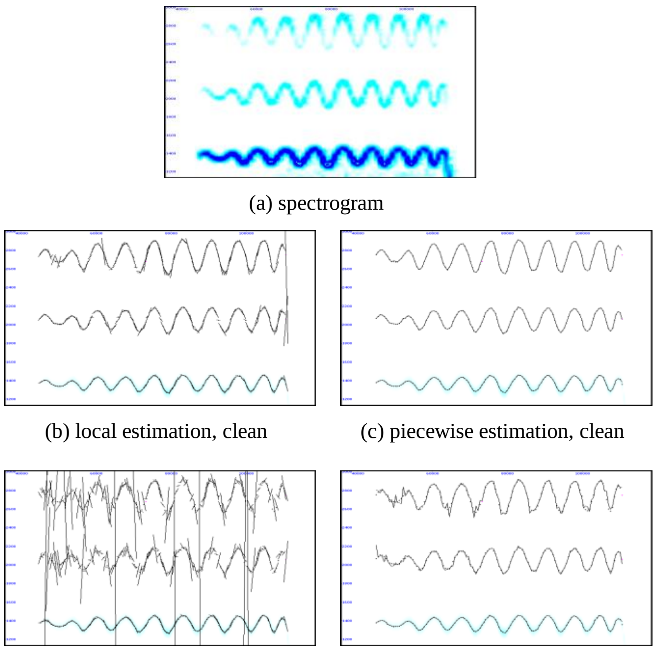
6. REAL-WORLD EXAMPLE

In this section we provide a real-world audio example using the two derivative algorithms to estimate frequencies and frequency slopes of sinusoidal partials.

We run the local and piecewise derivative algorithms on a pure vocal recording of a soprano singing /a:/. The spectrogram of the three lowest partials is given in Figure 5(a). We use linear frequency for the local derivative algorithm and linear spline frequency for the piecewise derivative algorithm.

For the three lowest partials we draw the frequency results in Figures 5(b) and 5(c). The local derivative algorithm estimates the frequency over individual frames as short line segments, each covering the duration of 2T. The piecewise derivative algorithm estimates it over the whole duration as a linear spline. While no frequency ground truth is available for real-world recordings, the disagreement of local estimates from adjacent frames provides an indication of the accuracy of the local derivative algorithm, while the smoothness of the frequency trajectories provides an indication of that of the piecewise algorithm. For the clean vocal signal both algorithms provide good frequency estimates.

Figures 5(d) and 5(e) shows the results when 0dB white noise is added to the signal. In the presence of this noise only the strongest fundamental partial retains good estimates. For the weaker partials both algorithms suffer performance loss. Although the drawings of Figs. 5(d) and 5(e) are not directly comparable, 5(e) does show more stable frequency slope than 5(d), in which the frequency slope errors are large enough to turn local frequency trajectories into near-vertical spikes. Since the piecewise algorithm relies on a global signal model, it is reasonable to expect higher noise tolerance, because the global constraints help to reduce the number of free parameters and prevent overfitting to local noise sources.



(a) spectrogram

(b) local estimation, clean

(c) piecewise estimation, clean

(d) local estimation, noisy

(e) piecewise estimation, noisy

Figure 5 . Local and piecewise frequency estimation of vocal /a:/

7. CONCLUSION

In this paper we have discussed the theory and implementation of a piecewise derivative algorithm, which approximates a slowly varying complex sinusoid as a spline exponential function. This algorithm is based on the recently developed distributive derivative algorithm, currently the state-of-the-art for non-stationary sinusoid estimation on short intervals. By adapting the derivative approach to piecewise models, we are able to estimate time-varying sinusoids of flexible length. The algorithm is simple, non-iterative, and more robust against noise than the (local) distributive derivative algorithm.

Apart from as the analyser of a sinusoidal modelling system, the proposed estimator may find applications wherever observation of sinusoidal parameters at a sequence of frames is required. For example, musicologists are interested in accurate tracing of vocal and instrumental vibratos and tremolos. In the frequency sweep test popular with audio, the proposed algorithm will allow faster sweep rate (since it does not assume stationarity) and will automatically return the result as a spline function.

The piecewise derivative algorithm has no source separation capacity, therefore does not solve the more difficult problem of estimating sinusoids overlapping other sinusoids or noise in both time and frequency. However, our tests show that the piecewise algorithm is less susceptible to concurrent sinusoids and noise than the local derivative algorithm.

Although we have focused our discussions on the spline exponential model, the piecewise derivative algorithm is also applicable to other linear piecewise parameterizations of the complex exponent, as long as they are compliant to (16).

8. ACKNOWLEDGEMENT

Part of this work was finished when the author was with Queen Mary, University of London and Professor Mark Sandler, with support from the Centre for Digital Music platform grant.

9. REFERENCES

[1] R.J. McAulay and T.F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol.34, no.4, 1986, pp.744-754.

[2] X. Serra, "Musical sound modeling with sinusoids plus noise," *Musical Signal Processing*, Swets & Zeitlinger Publishers, 1997, pp.91-122.

[3] Wen X. and M. Sandler, "On the characterization of slowly varying sinusoids," *EURASIP Journal on Audio, Speech and Music Processing*, vol. 2010, article 941732, 7 pages.

[4] F. Keiler and S. Marchand, "Survey on extraction of sinusoids in stationary sounds," in *Proc.5th International Conference on Digital Audio Effects (DAFx)*, Hamburg, 2002.

[5] P. Masri and N. Canagarajah, "Extracting more detail from the spectrum with phase distortion analysis," in *Proc. 1st COST-G6 Workshop on Digital Audio Effects (DAFx)*, Barcelona, 1998.

[6] A. Röbel, "Frequency slope estimation and its application for non-stationary sinusoidal parameter estimation," in *Proc. 10th International Conference on Digital Audio Effects (DAFx)*, Bordeaux, 2007.

[7] M. Abe and J. O. Smith III, "AM/FM rate estimation for time-varying sinusoidal modeling," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, Philadelphia, 2005.

[8] Wen X. and M. Sandler, "Notes on model-based non-stationary sinusoid estimation using derivatives," in *Proc. 12th International Conference on Digital Audio Effects (DAFx)*, Como, 2009.

[9] M. Betser, "Sinusoidal polynomial parameter estimation using the distribution derivative," *IEEE Transactions on Signal Processing*, vol.57 no.12, 2009.

[10] S. Muševič and J. Bonada, "Distribution derivative method for generalized sinusoid with complex amplitude modulation," in *Proc. 18th International Conference on Digital Audio Effects (DAFx)*, Trondheim, 2015.

[11] Y. Ding and X. Qian, "Processing of musical tones using a combined quadratic polynomial-phase sinusoid and residual (QUASAR) signal model," *Journal of the Audio Engineering Society*, vol.45, no.7/8, 1997.

[12] A. Röbel, "Adaptive additive modeling with continuous parameter trajectories," *IEEE Transactions on Audio, Speech, and Language Processing*, vol.14 no.4, 2006.

[13] M. Desainte-Catherine and S. Marchand, "High-precision Fourier analysis of sounds using signal derivatives," *Journal of the Audio Engineering Society*, vol.48 no.7/8, 2000.

[14] S. Marchand and P. Depalle, "Generalization of the derivative analysis method to non-stationary sinusoidal modeling," in *Proc.*

11th International Conference on Digital Audio Effects (DAFx), Espoo, 2008.

[15] M. Abe and J. O. Smith, "Design criteria for simple sinusoidal parameter estimation based on quadratic interpolation of FFT magnitude peaks," in *Proc. AES 117th Convention,* San Francisco, 2004.

10. APPENDIX: PIECEWISE MATRIX FORMULATIONS OF LINEAR, QUADRATIC AND CUBIC SPLINES

We denote a spline by $x(t)$ and its samples at the knot points by $\mathbf{x}=(x_0, \ldots, x_L)^{\mathrm{T}}$, $x_l=x(lT)$, $\forall l$. The linear interpolative formulation expresses $x(t)$ as linear functions of $x_0, \ldots, x_L$. Accordingly, the polynomial coefficients over $[lT, lT+T)$, i.e. $\boldsymbol{\lambda}_l$, can be given as a linear transformation of $\mathbf{x}$:

$$\boldsymbol{\lambda}_l=\mathbf{A}_l\mathbf{x},\ \forall l. \tag{A1}$$

10.1. Linear spline

A *linear spline* is specified by

$$\begin{cases} x(lT+t)=\lambda_{l,1}t+\lambda_{l,0},\ 0\le t<T,\ l=0,\cdots,L-1, \\ x(lT)=x_l,\ l=0,\cdots,L. \end{cases} \tag{A2}$$

It has the closed-form expression

$$x(lT+t)=x_l(1-t/T)+x_{l+1}\cdot t/T\ . \tag{A3}$$

For this spline type we can immediately write

$$\mathbf{A}_l=\begin{pmatrix}\mathbf{0}_{2\times l} & \begin{pmatrix}-1/T & 1/T \\ 1 & \end{pmatrix} & \mathbf{0}_{2\times(L-l-1)}\end{pmatrix}. \tag{A4}$$

10.2. Quadratic spline

A quadratic spline is specified by

$$\begin{cases} x(lT+t)=\lambda_{l,2}t^2+\lambda_{l,1}t+\lambda_{l,0},\ 0\le t<T,\ l=0,\cdots,L-1, \\ x(lT)=x_l,\ l=0,\cdots,L, \\ x'_-(lT)=x'_+(lT),\ l=1,\cdots,L-1. \end{cases} \tag{A5}$$

The standard approach for quadratic spline interpolation computes the polynomial coefficients from an intermediate vector $\mathbf{z}=(z_0, \ldots, z_{L-1})^{\mathrm{T}}$, $z_l=y'_l$, by

$$\begin{pmatrix}\lambda_{l,2} \\ \lambda_{l,1} \\ \lambda_{l,0}\end{pmatrix}=\begin{pmatrix}-1/T \\ 1 \\ 0\end{pmatrix}z_l+\begin{pmatrix}-1/T^2 & 1/T^2 \\ 0 & 0 \\ 1 & 0\end{pmatrix}\begin{pmatrix}x_l \\ x_{l+1}\end{pmatrix} \tag{A6}$$

or

$$\boldsymbol{\lambda}_l=(\mathbf{0}_{3\times l}\ \ \mathbf{A}_{\mathrm{Z}}\ \ \mathbf{0}_{3\times(L-l-1)})\mathbf{z}+(\mathbf{0}_{3\times l}\ \ \mathbf{A}_{\mathrm{C}}\ \ \mathbf{0}_{3\times(L-l-1)})\mathbf{x}\ , \tag{A7}$$

where $\mathbf{A}_{\mathrm{Z}}$ and $\mathbf{A}_{\mathrm{C}}$ replace the two matrices in (A6) for briefness. $\mathbf{z}$ is related to $\mathbf{x}$ by the linear system

$$\begin{pmatrix}1 & 1 & & & \\ & 1 & 1 & & \\ & & \cdots & \cdots & \cdots \\ & & & 1 & 1\end{pmatrix}\mathbf{z}=\frac{2}{T}\begin{pmatrix}-1 & 1 & & & 0 \\ & -1 & 1 & & 0 \\ & & \cdots & \cdots & \cdots & 0 \\ & & & -1 & 1 & 0\end{pmatrix}\mathbf{x}. \tag{A8}$$

Eq.(A8) is underdetermined by one equation. We introduce an additional constraint by minimizing the square norm of the spline's derivative, i.e.

$$I=\int_0^{LT}\left(x'(t)\right)^2dt=\frac{T}{3}\sum_{l=0}^{L-1}\left(z_l^2-2z_l\frac{x_{l+1}-x_l}{T}+4\frac{(x_{l+1}-x_l)^2}{T^2}\right). \tag{A9}$$

We call the specific quadratic spline that minimizes (A9) the minimal-variation quadratic spline. Let $dI/d\mathbf{z}=0$, we get a new linear equation

$$T\sum_{l=0}^{L-1}(-1)^l z_l+x_0+2\sum_{l=1}^{L-1}(-1)^l x_l+(-1)^L x_L=0\ . \tag{A10}$$

Eq.(A10) pads up the matrix on the left of (A8) to $L\times L$ and the on the right to $L\times(L+1)$, after which (A8) takes the form $\mathbf{M}_L\mathbf{z}=\mathbf{M}_R\mathbf{x}$, $\mathbf{M}_L$ being square and invertible. Substituting $\mathbf{z}=\mathbf{M}_L^{-1}\mathbf{M}_R\mathbf{x}$ into (A7) we get

$$\boldsymbol{\lambda}_l=((\mathbf{0}_{3\times l}\ \ \mathbf{A}_{\mathrm{Z}}\ \ \mathbf{0}_{3\times(L-l-1)})\mathbf{M}_L^{-1}\mathbf{M}_R+(\mathbf{0}_{3\times l}\ \ \mathbf{A}_{\mathrm{C}}\ \ \mathbf{0}_{3\times(L-l-1)}))\mathbf{x}\ . \tag{A11}$$

This gives the formulation of $\mathbf{A}_l$ in (A1) as

$$\mathbf{A}_l=(\mathbf{0}_{3\times l}\ \ \mathbf{A}_{\mathrm{Z}}\ \ \mathbf{0}_{3\times(L-l-1)})\mathbf{M}_L^{-1}\mathbf{M}_R+(\mathbf{0}_{3\times l}\ \ \mathbf{A}_{\mathrm{C}}\ \ \mathbf{0}_{3\times(L-l-1)}). \tag{A12}$$

10.3. Cubic spline

A cubic spline is specified by

$$\begin{cases} x(lT+t)=\lambda_{l,3}t^3+\lambda_{l,2}t^2+\lambda_{l,1}t+\lambda_{l,0},\ 0\le t<T,\ l=0,\cdots,L-1, \\ x(lT)=x_l,\ l=0,\cdots,L, \\ x'_-(lT)=x'_+(lT), x''_-(lT)=x''_+(lT),\ l=1,\cdots,L-1. \end{cases} \tag{A13}$$

The standard approach for cubic spline interpolation computes the polynomial coefficients from an intermediate vector $\mathbf{z}=(z_0,\ldots, z_L)^{\mathrm{T}}$, $z_l=y''_l$, by

$$\begin{pmatrix}\lambda_{l,3} \\ \lambda_{l,2} \\ \lambda_{l,1} \\ \lambda_{l,0}\end{pmatrix}=\begin{pmatrix}-1/6T & 1/6T \\ 1/2 & 0 \\ -T/3 & -T/6 \\ 0 & 0\end{pmatrix}\begin{pmatrix}z_l \\ z_{l+1}\end{pmatrix}+\begin{pmatrix}0 & 0 \\ 0 & 0 \\ -1/T & 1/T \\ 1 & 0\end{pmatrix}\begin{pmatrix}x_l \\ x_{l+1}\end{pmatrix} \tag{A14}$$

or

$$\boldsymbol{\lambda}_l=\begin{pmatrix}\mathbf{0}_{4\times l} & \mathbf{A}_{\mathrm{Z}} & \mathbf{0}_{4\times(L-l-1)}\end{pmatrix}\mathbf{z}+\begin{pmatrix}\mathbf{0}_{4\times l} & \mathbf{A}_{\mathrm{C}} & \mathbf{0}_{4\times(L-l-1)}\end{pmatrix}\mathbf{x} \tag{A15}$$

where $\mathbf{A}_{\mathrm{Z}}$ and $\mathbf{A}_{\mathrm{C}}$ replace the two matrices in (A14) for briefness. $\mathbf{z}$ is related to $\mathbf{x}$ by the linear system

$$\begin{pmatrix}1 & 4 & 1 & & \\ & 1 & 4 & 1 & \\ & & \cdots & \cdots & \cdots \\ & & & 1 & 4 & 1\end{pmatrix}\mathbf{z}=\frac{6}{T^2}\begin{pmatrix}1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & \cdots & \cdots & \cdots \\ & & & 1 & -2 & 1\end{pmatrix}\mathbf{x}. \tag{A16}$$

This is an underdetermined system. Two more constraints are needed to uniquely specify $\mathbf{z}$ from $\mathbf{x}$. The constraints typically concern the behaviour of $\mathbf{z}$ near 0 and L, referred to as the boundary mode, e.g.

a) natural mode: $z_0=z_L=0$;
b) quadratic run-out mode: $z_0=z_1$, $z_L=z_{L-1}$;
c) cubic run-out mode: $z_0=2z_1\text{-}z_2$, $z_L=2z_{L-1}\text{-}z_{L-2}$.

Whichever mode we choose, the constraints pad up the matrices on both sides of (A16) to $(L+1)\times(L+1)$, after which (A16) takes the form $\mathbf{M}_L\mathbf{z}=\mathbf{M}_R\mathbf{x}$, $\mathbf{M}_L$ being invertible. Substituting $\mathbf{z}=\mathbf{M}_L^{-1}\mathbf{M}_R\mathbf{x}$ into (A15) we get

$$\boldsymbol{\lambda}_l=((\mathbf{0}_{4\times l}\ \ \mathbf{A}_{\mathrm{Z}}\ \ \mathbf{0}_{4\times(L-l-1)})\mathbf{M}_L^{-1}\mathbf{M}_R+(\mathbf{0}_{4\times l}\ \ \mathbf{A}_{\mathrm{C}}\ \ \mathbf{0}_{4\times(L-l-1)}))\mathbf{x}. \tag{A17}$$

This gives the formulation of $\mathbf{A}_l$ in (A1) as

$$\mathbf{A}_l=(\mathbf{0}_{4\times l}\ \ \mathbf{A}_{\mathrm{Z}}\ \ \mathbf{0}_{4\times(L-l-1)})\mathbf{M}_L^{-1}\mathbf{M}_R+(\mathbf{0}_{4\times l}\ \ \mathbf{A}_{\mathrm{C}}\ \ \mathbf{0}_{4\times(L-l-1)}). \tag{A18}$$