

Enhanced Demodulation for Low Latency Audio Pitch Shifting in Frequency Domain

Supplementary notebook for equation check

In this Wolfram Mathematica notebook, the equations from paper “Enhanced Demodulation for Low Latency Audio Pitch Shifting in Frequency Domain” are automatically checked. Figure 1 is also generated.

As Mathematica uses some variables beginning with upper-case Latin letters for its internal purposes, variable “N” from paper is represented by “vN” and variable “O” from the paper is represented by “vO” internally.

Output with general frame overlaps (Subsection 4.1)

General sinusoid equation, centered on bin a of vN sample DFT (inline equation in subsection 4.1) and its exponential representation (Equation 3).

We store the exponential forms without the outside sums applied, applying them only at the end. The forms are compared for equality (the output should be “True”).

```
In[142]:= inputSinusoidTrigForm = A * Cos[2 * π *  $\frac{n * a}{vN}$  + φ];  
  
inputSinusoid =  $\frac{A}{2} * e^{i * q * (2 * \pi * \frac{n * a}{vN} + \phi)}$ ;  
  
Equal[TrigToExp[inputSinusoidTrigForm], ExpandAll[Sum[inputSinusoid, {q, -1, 1, 2}]]]  
  
Out[144]= True
```

Running Hanning window in trigonometric and exponential form (Equation 4).

The forms are compared for equality (the output should be “True”).

```
In[145]:= runningHanningWindowTrigForm =  $\frac{1}{2} - \frac{1}{2} \cos[2 * \pi * (\frac{n}{vN} + \frac{p}{vO})]$ ;  
  
runningHanningWindow =  $\frac{1}{4} - \frac{1}{4} e^{i * 2 * \pi * r * (\frac{n}{vN} + \frac{p}{vO})}$ ;  
  
Equal[ExpandAll[TrigToExp[runningHanningWindowTrigForm]],  
ExpandAll[Sum[runningHanningWindow, {r, -1, 1, 2}]]]  
  
Out[147]= True
```

The “computed form” of Equation 5 is found by multiplying the trigonometric forms of Equation 3 and Equation 4.

The first exponential form used in Equation 5 is also provided.

The forms are compared for equality (the output should be “True”).

```

In[148]:= analysedResultTrigForm = inputSinusoidTrigForm * runningHanningWindowTrigForm;
analysedResult =  $\frac{A}{8} * \left( e^{i * q * \left( 2 \pi \frac{a * n}{vN} + \phi \right)} - e^{i * \left( 2 \pi \left( \frac{(q * a + r) * n}{vN} + \frac{r * p}{v0} \right) + q * \phi \right)} \right);$ 
ExpToTrig[ExpandAll[Sum[Sum[analysedResult, {q, -1, 1, 2}], {r, -1, 1, 2}]]]
Equal[ExpandAll[TrigToExp[analysedResultTrigForm]],
ExpandAll[Sum[Sum[analysedResult, {q, -1, 1, 2}], {r, -1, 1, 2}]]]
Out[150]=  $-\frac{1}{4} A \cos \left[ \frac{2 n \pi}{vN} - \frac{2 a n \pi}{vN} + \frac{2 p \pi}{v0} - \phi \right] + \frac{1}{2} A \cos \left[ \frac{2 a n \pi}{vN} + \phi \right] - \frac{1}{4} A \cos \left[ \frac{2 n \pi}{vN} + \frac{2 a n \pi}{vN} + \frac{2 p \pi}{v0} + \phi \right]$ 
Out[151]= True

```

The second exponential form used in Equation 5 is compared for equality here.

```

In[152]:= alternativeAnalysedResult =  $\frac{A}{8} * \left( e^{i * q * \left( 2 \pi \frac{a * n}{vN} + \phi \right)} - e^{i * q * \left( 2 \pi \left( \frac{(a + r) * n}{vN} + \frac{r * p}{v0} \right) + \phi \right)} \right);$ 
Equal[ExpandAll[Sum[Sum[analysedResult, {q, -1, 1, 2}], {r, -1, 1, 2}]],
ExpandAll[Sum[Sum[alternativeAnalysedResult, {q, -1, 1, 2}], {r, -1, 1, 2}]]]
Out[153]= True

```

Now, the running window should be shifted (Equation 6). We introduce a function $g(h)$ which shifts the bins and function $u(\phi)$ which shifts the phase of the input sinusoid.

The $\frac{r * p}{v0}$ part is not shifted due to the phase shift counteraction in the bin translation routine, which should ensure it stays as it was.

When these functions are identities, the “shifted” running window should be equal to that in Equation 5 (the output should be “True”).

```

In[154]:= shiftedResult =  $\frac{A}{8} * \left( e^{i * q * \left( 2 \pi * \frac{g[a] * n}{vN} + u[\phi] \right)} - e^{i * q * \left( 2 \pi * \left( \frac{g[a + r] * n}{vN} + \frac{r * p}{v0} \right) + u[\phi] \right)} \right);$ 
Equal[Sum[Sum[alternativeAnalysedResult, {q, -1, 1, 2}], {r, -1, 1, 2}],
Sum[Sum[shiftedResult, {q, -1, 1, 2}], {r, -1, 1, 2}] /. {g[h_] -> h, u[phi_] -> phi}]
Out[155]= True

```

The output signal equation (Equation 7) is now checked. The synthesis window is the same as the analysis window, we just change the summing variable.

```

In[156]:= oneFrameSynthesisResult =
ExpandAll[FullSimplify[shiftedResult * (runningHanningWindow /. {r -> s})]];
oneFrameSynthesisResultPaper =  $\frac{A}{32} * \left( e^{i * q * \left( 2 \pi * \frac{g[a] * n}{vN} + u[\phi] \right)} - e^{i * q * \left( 2 \pi * \left( \frac{g[a + r] * n}{vN} + \frac{r * p}{v0} \right) + u[\phi] \right)} - \right.$ 
 $e^{i * \left( 2 \pi * \left( \frac{(q * g[a] + s) * n}{vN} + \frac{s * p}{v0} \right) + q * u[\phi] \right)} + e^{i * \left( 2 \pi * \left( \frac{(q * g[a + r] + s) * n}{vN} + \frac{(q * r + s) * p}{v0} \right) + q * u[\phi] \right)} \left. \right);$ 
oneFrameSynthesisResultExpanded = ExpandAll[
Sum[Sum[Sum[oneFrameSynthesisResult, {q, -1, 1, 2}], {r, -1, 1, 2}], {s, -1, 1, 2}]];
oneFrameSynthesisResultPaperExpanded = ExpandAll[Sum[Sum[
Sum[oneFrameSynthesisResultPaper, {q, -1, 1, 2}], {r, -1, 1, 2}], {s, -1, 1, 2}]];
Equal[oneFrameSynthesisResultExpanded, oneFrameSynthesisResultPaperExpanded]
Out[160]= True

```

Output with reasonable frame overlaps (Subsection 4.2)

We simply sum the output signal equations over appropriate overlaps. For frame overlap that is a multiple of 2, we sum $p = 0$ and $p = \frac{v_0}{2}$. For frame overlap that is a multiple of 4, we sum this result with its copy that is $\frac{v_0}{4}$ frames apart.

We then perform the sum and perform full simplification. This should give three cosine terms. The one that is multiplied by 4 should be the wanted one. The other terms are unwanted and result in modulation artifacts.

Note that the resulting expression no longer depends on frame number p , meaning that the result no longer depends on frame numbers. Therefore, they can just be summed afterwards.

We check that the result is equal to Equation 8.

```
In[161]:= twoFrameSynthesisResult =
  oneFrameSynthesisResult +  $\left( \text{oneFrameSynthesisResult} /. \{p \rightarrow p + \frac{v_0}{2}\} \right);$ 
fourFrameSynthesisResult = twoFrameSynthesisResult +
   $\left( \text{twoFrameSynthesisResult} /. \{p \rightarrow p + \frac{v_0}{4}\} \right);$ 
fourFrameOutput = FullSimplify[Sum[
  Sum[Sum[fourFrameSynthesisResult, {q, -1, 1, 2}], {r, -1, 1, 2}], {s, -1, 1, 2}]];
basicAlgorithmOutput = (v0/4) * fourFrameOutput;
basicAlgorithmOutputPaper =  $\frac{A * v_0}{16} * \left( 4 * \text{Cos}\left[2 \pi * \frac{n}{vN} * g[a] + u[\phi]\right] + \right.$ 
 $\left. \text{Cos}\left[2 \pi * \frac{n}{vN} * (g[a - 1] + 1) + u[\phi]\right] + \text{Cos}\left[2 \pi * \frac{n}{vN} * (g[a + 1] - 1) + u[\phi]\right] \right);$ 
Equal[basicAlgorithmOutput, basicAlgorithmOutputPaper]

Out[166]= True
```

Exact Ocean algorithm modulation curve (Subsection 4.3)

To check the exact Ocean algorithm modulation curve in Equation 9, we simply apply the definitions of the functions $g(x)$ and $u(x)$ as used in the Ocean algorithm and divide the actual output by the expected output.

```

In[167]:= oceanAlgorithmRules = {g[a_] → Floor[a * k + 1/2], u[φ_] → φ};

actualOceanOutput = basicAlgorithmOutput /. oceanAlgorithmRules;
expectedOutput = A * Cos[ $\frac{2 n \pi g[a]}{vN} + u[\phi]$ ];
expectedOceanOutput = expectedOutput /. oceanAlgorithmRules;
oceanModulationCurve = actualOceanOutput / expectedOceanOutput;
oceanModulationCurvePaper =  $\frac{v0}{16} * \left( 4 * \text{Cos}\left[2 \pi * \frac{n}{vN} * \text{Floor}[a * k + 1/2] + \phi\right] + \right.$ 
 $\text{Cos}\left[2 \pi * \frac{n}{vN} * (\text{Floor}[(a - 1) * k + 1/2] + 1) + \phi\right] + \text{Cos}\left[2 \pi * \frac{n}{vN} * (\text{Floor}[(a + 1) * k + 1/2] - 1) + \phi\right]$ 
 $\left. \right) / \text{Cos}\left[2 \pi * \frac{n}{vN} * \text{Floor}[a * k + 1/2] + \phi\right];$ 
Equal[oceanModulationCurve, oceanModulationCurvePaper]

Out[173]= True

```

If the scaling factor k is an integer, the curve simplifies and becomes independent on actual bin number a (Equation 10). If it is equal to one, the curve is constant, as would be expected.

```

In[174]:= oceanIntegerPitchMultiplierCurve = FullSimplify[oceanModulationCurve, k ∈ Integers]
oceanIntegerPitchMultiplierCurve /. {k → 1}

Out[174]=  $\frac{1}{8} v0 \left( 2 + \text{Cos}\left[\frac{2 (-1 + k) n \pi}{vN}\right] \right)$ 

Out[175]=  $\frac{3 v0}{8}$ 

```

For scaling factor $k = 1.5$, dependence on bin number a is apparent even when odd and even bins are considered separately (Equation 11).

```

In[176]:= oneAndHalfEvenBinsModulationCurve = FullSimplify[FullSimplify[
  (oceanModulationCurve) /. {k → 3/2, a → 2 * m}, m ∈ Integers] /. {m → a/2}];
oneAndHalfEvenBinsModulationCurvePaper =
 $\frac{v0}{16} * \left( 5 + \text{Cos}\left[\pi * \frac{n}{vN} * (3 a + 2) + \phi\right] / \text{Cos}\left[\pi * \frac{n}{vN} * 3 a + \phi\right] \right);$ 
Equal[oneAndHalfEvenBinsModulationCurve, oneAndHalfEvenBinsModulationCurvePaper]

Out[178]= True

In[179]:= oneAndHalfOddBinsModulationCurve =
  FullSimplify[FullSimplify[(oceanModulationCurve) /. {k → 3/2, a → (2 * m + 1)},
    m ∈ Integers] /. {m → (a - 1)/2}];
oneAndHalfOddBinsModulationCurvePaper =
 $\frac{v0}{16} * \left( 5 + \text{Cos}\left[\pi * \frac{n}{vN} * (3 a - 1) + \phi\right] / \text{Cos}\left[\pi * \frac{n}{vN} * (3 a + 1) + \phi\right] \right);$ 
Equal[ExpandAll[oneAndHalfOddBinsModulationCurve],
  ExpandAll[oneAndHalfOddBinsModulationCurvePaper]]

Out[181]= True

```

Proposed inexact demodulation curve (Subsection 5.1)

Not much can be done with the precise modulation curve as it is heavily dependent on the bin number a . We choose a curve that follows the actual modulation curve reasonably well.

$$\begin{aligned}
 \text{In[182]:= } \text{ourCurve} &= \frac{1}{16} v_0 \left(4 + \cos \left[2 \pi * \frac{n}{vN} \left(\text{Floor} \left[(a-1) k + \frac{1}{2} \right] - \text{Floor} \left[a k + \frac{1}{2} \right] + 1 \right) \right] + \right. \\
 &\quad \left. \cos \left[2 \pi * \frac{n}{vN} \left(\text{Floor} \left[(a+1) k + \frac{1}{2} \right] - \text{Floor} \left[a k + \frac{1}{2} \right] - 1 \right) \right] \right); \\
 \text{ourDemodulatedOutput} &= \text{FullSimplify}[\text{actualOceanOutput} / \text{ourCurve}] \\
 \text{Out[183]= } &\left(A \left(\cos \left[\phi + \frac{2 n \pi \left(1 + \text{Floor} \left[\frac{1}{2} + (-1+a) k \right] \right)}{vN} \right] + \right. \right. \\
 &\quad \left. \left. 4 \cos \left[\phi + \frac{2 n \pi \text{Floor} \left[\frac{1}{2} + a k \right]}{vN} \right] + \cos \left[\phi + \frac{2 n \pi \left(-1 + \text{Floor} \left[\frac{1}{2} + k + a k \right] \right)}{vN} \right] \right) \right) / \\
 &\left(4 + \cos \left[\frac{2 n \pi \left(1 + \text{Floor} \left[\frac{1}{2} + (-1+a) k \right] - \text{Floor} \left[\frac{1}{2} + a k \right] \right)}{vN} \right] + \right. \\
 &\quad \left. \cos \left[\frac{2 n \pi \left(1 + \text{Floor} \left[\frac{1}{2} + a k \right] - \text{Floor} \left[\frac{1}{2} + k + a k \right] \right)}{vN} \right] \right)
 \end{aligned}$$

Our curve simplifies to the same thing as the exact curve if the pitch multiplier k is integer.

$$\begin{aligned}
 \text{In[184]:= } &\text{FullSimplify}[\text{ourCurve}, k \in \text{Integers}] \\
 &\text{FullSimplify}[\text{ourCurve} /. \{k \rightarrow 1\}, a \in \text{Integers}] \\
 \text{Out[184]= } &\frac{1}{8} v_0 \left(2 + \cos \left[\frac{2 (-1+k) n \pi}{vN} \right] \right) \\
 \text{Out[185]= } &\frac{3 v_0}{8}
 \end{aligned}$$

In case $a = 0$ and the fractional part of k is not exactly $\frac{1}{2}$, the proposed inexact equation and the exact output equation become equivalent. This is due to the properties of the floor function and hard to prove purely in Mathematica.

It is, however, somewhat apparent from the fact that if the fractional part is not exactly $\frac{1}{2}$, the two cosine functions below will have the same, just negated, arguments.

```

In[186]:= FullSimplify[oceanModulationCurvePaper /. {a -> 0}]
FullSimplify[ourCurve /. {a -> 0}]
FullSimplify[oceanModulationCurvePaper /. {a -> 0, k -> 2.1}]
FullSimplify[ourCurve /. {a -> 0, k -> 2.1}]
FullSimplify[oceanModulationCurvePaper /. {a -> 0, k -> 2.5}]
FullSimplify[ourCurve /. {a -> 0, k -> 2.5}]

Out[186]= 
$$\frac{1}{16} v_0 \left( 4 \cos[\phi] + \cos\left[\phi + \frac{2 n \pi \left(1 + \text{Floor}\left[\frac{1}{2} - k\right]\right)}{vN}\right] + \cos\left[\phi + \frac{2 n \pi \left(-1 + \text{Floor}\left[\frac{1}{2} + k\right]\right)}{vN}\right] \right) \sec[\phi]$$


Out[187]= 
$$\frac{1}{16} v_0 \left( 4 + \cos\left[\frac{2 n \pi \left(1 + \text{Floor}\left[\frac{1}{2} - k\right]\right)}{vN}\right] + \cos\left[\frac{2 n \pi \left(-1 + \text{Floor}\left[\frac{1}{2} + k\right]\right)}{vN}\right] \right)$$


Out[188]= 
$$\frac{1}{8} v_0 \left( 2 + \cos\left[\frac{2 n \pi}{vN}\right] \right)$$


Out[189]= 
$$\frac{1}{8} v_0 \left( 2 + \cos\left[\frac{2 n \pi}{vN}\right] \right)$$


Out[190]= 
$$\frac{1}{16} v_0 \left( \cos\left[\frac{2 n \pi}{vN} - \phi\right] + 4 \cos[\phi] + \cos\left[\frac{4 n \pi}{vN} + \phi\right] \right) \sec[\phi]$$


Out[191]= 
$$\frac{1}{16} v_0 \left( 4 + \cos\left[\frac{2 n \pi}{vN}\right] + \cos\left[\frac{4 n \pi}{vN}\right] \right)$$


```

Error introduced by the inexact modulation curve (Subsection 5.3)

We compute the error introduced by the inexact modulation curve. We use the knowledge that there can be only two versions of the inexact modulation curve (Subsection 5.2 of the paper) and that the “symmetrical” modulation curve version produces exact results.

This means that we only need to find the error for the “asymmetrical” modulation curve version. The error depends on the expected sinusoid, which can be thought of as determined by the bin number a for the purposes of error computation, and the actual sinusoid which, in addition to the bin number a , also depends on whole part of the scaling factor k .

To compute the error introduced, we take the equations for expected output and actual output demodulated by our inexact curve. We then select the asymmetrical modulation curve version. To obtain power equations, we multiply both the expected output and actual output by the actual output. This produces two results: the equation for power in the wanted bin and equation for total power.

The actual power values are obtained by numerical integration over desired whole parts of k and bins. The values are only computed for bins that may correspond to the asymmetrical modulation curve version.

The final power of unwanted spectral components is then computed and converted to decibels. As power is being computed, the $10 \cdot \log_{10}(x)$ version of dB computation formula is used.

```

In[192]:= (* Select the asymmetrical curve version *)
floorReplacers = {Floor[ $\frac{1}{2} + (-1 + a) k$ ] → floorMinus,
  Floor[ $\frac{1}{2} - k + a k$ ] → floorMinus, Floor[ $\frac{1}{2} + a k$ ] → floorZero,
  Floor[ $\frac{1}{2} + (1 + a) k$ ] → floorPlus, Floor[ $\frac{1}{2} + k + a k$ ] → floorPlus};
{ (expectedOceanOutput /. {ϕ → 0, vN → 1}), (ourDemodulatedOutput /. {ϕ → 0, vN → 1}) } /.
  {A → 1} /. floorReplacers;
FullSimplify[% /. {floorMinus → floorZero - floorK, floorPlus → floorZero + floorK + 1}];

(* Obtain power equations *)
powerEquations = FullSimplify[%*%[[2]]]
Out[195]= { (Cos[2 floorZero n π] (4 Cos[2 floorZero n π] +
  Cos[2 (1 - floorK + floorZero) n π] + Cos[2 (floorK + floorZero) n π])) /
  (4 + Cos[2 (-1 + floorK) n π] + Cos[2 floorK n π]),
  (4 Cos[2 floorZero n π] + Cos[2 (1 - floorK + floorZero) n π] +
  Cos[2 (floorK + floorZero) n π])2 / (4 + Cos[2 (-1 + floorK) n π] + Cos[2 floorK n π])2 }

In[196]:= (* Function for computation of the relative errors *)
computeErrors[k_] :=
Block[{consideredTable, consideredBins, consideredPowers, selectedConsideredBins},
  (consideredTable = Table[{k * mult, (k + 1) * mult}, {mult, 1, 50}];
  consideredBins = DeleteDuplicates[
    Flatten[Map[Table[bin, {bin, #[[1]], #[[2]]}] &, consideredTable]]];
  selectedConsideredBins = Cases[consideredBins, x_ /; x ≤ 15];
  consideredPowers =
    Map[{#, NIntegrate[powerEquations /. {floorK → k, floorZero → #}, {n, 0, 1}]} &,
      selectedConsideredBins];
  Map[{#[[1]], 10 * Log10[(#[[2, 2]] - #[[2, 1]]) / #[[2, 2]]]} &, consideredPowers]]]

(* For whole parts 0 and 1, the error is the same, thus,
scaling factor 0 does not need to be computed explicitly. *)
Equal[powerEquations /. {floorK → 0}, powerEquations /. {floorK → 1}]
(* For the graph, we compute errors for whole parts 1,
2, and 3; higher ones are not very likely to be used *)
computedErrors = Table[computeErrors[k], {k, 1, 3}]

Out[197]= True

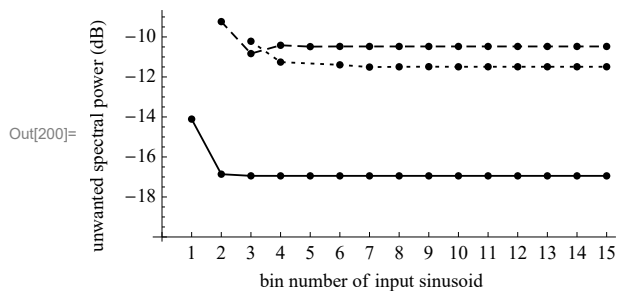
Out[198]= {{ {1, -14.1073}, {2, -16.8614}, {3, -16.9443}, {4, -16.9456}, {5, -16.9456},
  {6, -16.9456}, {7, -16.9456}, {8, -16.9456}, {9, -16.9456}, {10, -16.9456},
  {11, -16.9456}, {12, -16.9456}, {13, -16.9456}, {14, -16.9456}, {15, -16.9456}},
  {{2, -9.23382}, {3, -10.8324}, {4, -10.4168}, {5, -10.4843}, {6, -10.4745},
  {7, -10.4757}, {8, -10.4756}, {9, -10.4756}, {10, -10.4756}, {11, -10.4756},
  {12, -10.4756}, {13, -10.4756}, {14, -10.4756}, {15, -10.4756}},
  {{3, -10.2164}, {4, -11.2576}, {6, -11.3948}, {7, -11.5071},
  {8, -11.4933}, {9, -11.487}, {10, -11.492}, {11, -11.4904},
  {12, -11.4906}, {13, -11.4907}, {14, -11.4906}, {15, -11.4906}} }

```

```

In[199]:= (* Figure 1 is created by plotting the obtained values *)
thick = Thickness[0.004];
inexactPowerImage = Labeled[ListPlot[computedErrors,
  Joined → True, Mesh → Full, MeshStyle → Directive[PointSize[0.016], Black],
  PlotStyle → {{Black, thick}, {Dashed, Black, thick}, {Dotted, Black, thick},
    {DotDashed, Black, thick}}, AxesOrigin → {0, -20}, ImageSize → 256,
  AxesStyle → Black, LabelStyle → {FontSize → 10, FontFamily → "Times", Black},
  AspectRatio → 1/2, Ticks → {Table[x, {x, 0, 20}], Automatic}],
  {"bin number of input sinusoid", "unwanted spectral power (dB)"},
  {Bottom, Left}, RotateLabel → True,
  LabelStyle → {FontSize → 10, FontFamily → "Times", Black}]
SetDirectory[NotebookDirectory[]];
Export["inexact_power.pdf", inexactPowerImage]

```



Out[202]= inexact_power.pdf