FAST ANALYSIS/SYNTHESIS OF HARMONIC SIGNALS

Miltiadis Vasilakis, Yannis Agiomyrgiannakis and Yannis Stylianou

Department of Computer Science, University of Crete, Hellas and
Institute of Computer Science, FORTH
{mvasilak, jagiom, yannis}@csd.uoc.gr

ABSTRACT

Harmonic Models are commonly used in signal processing. The analysis of harmonic signals requires the solution of a symmetric Toeplitz system of equations. Levinson-based Toeplitz solvers have a $O(n^2)$ complexity. This paper proposes an $O(n)$ algorithm by encoding the inverse matrices required for the solution of the linear system to a few parameters in order to obtain an approximate solution for the harmonic model. For speech related applications, the proposed algorithm is 2-30 times faster than the Levinson algorithm, while degradation is minimal and memory requirements are very low.

1. INTRODUCTION

Harmonic representations of signals provide a valuable tool for signal processing and there are many methods that use or imply a Harmonic Model (HM) for the representation of a signal space. Some of the most important applications of HM are related to speech processing and the speech processing community has extensively used the fact that the speech signal exhibits a (quasi)harmonic behavior in voiced parts. Harmonic models of speech have been successfully used in speech coding, speech analysis/synthesis, voice transformation, text-to-speech, etc.

Sampling the spectrum of a windowed discrete signal at harmonic frequencies: $\omega_k = k\omega_0,\ k = 1, ..., K$, where K is the maximum number of harmonics, is an expensive operation when the harmonic frequencies do not coincide with the FFT frequency bins. Let $x(n)$ be the signal and $w(n)$, $n = -N, ..., N$ be the window applied to the signal. The analysis process in HM is actually the projection of the windowed signal $x_w(n) = w(n)x(n)$ into the subspace generated by the windowed harmonically related cosines and sines:

$$\begin{aligned} c_k(n) &= w(n)\cos(k\omega_0 n) \\ s_k(n) &= w(n)\sin(k\omega_0 n) \end{aligned} \tag{1}$$

The projection involves the computation of a pseudoinverse or equivalently the solution of a real symmetric Toeplitz system of equations with arbitrary right hand side. The most efficient symmetric Toeplitz solvers for matrix dimensions of less than one hundred belong to the class of Levinson-type algorithms. Levinson's algorithm [1] has a complexity of $4n^2 + O(n)$ flops. An improvement to a complexity of $3n^2 + O(n)$ flops was made in [2] with the *split-Levinson* algorithm. A further improvement was made with the *even-odd split Levinson* algorithm [3] which has a complexity of $\frac{5}{2}n^2 + O(n)$ flops. A recent publication [4] introduces a relatively small improvement for certain cases. However, all these methods require $O(n^2)$ complexity to provide an exact solution.

In this paper we present approximate solutions to the problem of fast HM analysis and synthesis. For analysis, our approximate solution requires a complexity of $O(n)$. The basic idea is to exploit the patterns that appear inside the HM inverse cosine and sine correlation matrices (A_c^{-1} and A_s^{-1} respectively) in order to *encode* these matrices for a set of values in interest. Note that the A_c^{-1} and A_s^{-1} matrices are a function of three parameters: ω_0, $w(n)$ and the sampling rate F_s. Practically, the only free variable for HM analysis with fixed window is ω_0. Therefore, instead of *computing* the inverse matrices for specific ω_0 we can simply *decode* them. Synthesis of harmonically related sinusoids is also a time consuming operation. In [5] the computation of the cosine and sine bases is accelerated using lookup tables and recurrence relations between the sinusoids. However, the lookup table method requires an expensive modulo division operation. We propose a new set of recurrence relations that constructs the cosine and sine harmonic bases with a minimal cost of approximately 1 MAC (Multiply-Accumulate) instruction per sample of the bases vectors.

In Section 2 we provide the necessary background and introduce the notation. The algorithm that encodes A_c^{-1} and A_s^{-1} is presented in Section 3. Section 4 proposes a fast way to compute the cosine and sine bases vectors which are needed to construct the right hand side of the linear systems. The proposed analysis method is then evaluated in Section 5 in terms of SNR. Furthermore, speed comparisons between the proposed method and the standard Levinson algorithm indicate a significant improvement. Section 6 concludes the paper.

2. BACKGROUND

The Harmonic Model is a parametric model used for signal analysis/synthesis. The signal is represented as a weighted sum of harmonic cosines and sines:

$$\hat{x}(n) = \sum_{k=1}^{K} [c_k \cos(k\omega_o n) + s_k \sin(k\omega_o n)] \tag{2}$$

where ω_o is the fundamental frequency, K is the number of the harmonics, c_k and s_k are the cosine and sine coefficients describing the even and odd part of the k-th harmonic sinusoid respectively, and n is the time index. The unknown parameters c_k and s_k are evaluated using a weighted least-squares method that minimizes the square error criterion with respect to c_k and s_k:

$$\epsilon = \sum_{n=-N}^{N} w^2(n)(x(n) - \hat{x}(n))^2 \tag{3}$$

where $x(n)$ is the original signal, $\hat{x}(n)$ is its harmonic representation, $w(n)$ is a symmetric window and $2N + 1$ is the duration of the analysis frame.

Using matrix formulation, we may rewrite (2) as

$$\hat{x} = B \begin{bmatrix} c \\ s \end{bmatrix} \tag{4}$$

where B is the $(2N+1) - by - 2K$ cosine/sine basis matrix

$$B = [C\ S] \tag{5}$$

and where C and S are the cosine and sine bases matrices, respectively, with size $(2N+1) - by - K$ and elements that are defined by:

$$C_{n,k} = \cos(k\omega_o n) \tag{6}$$

$$S_{n,k} = \sin(k\omega_o n) \tag{7}$$

for $n = -N, \ldots, N$ and $k = 1, \ldots, K$, while vectors c, s hold the parameters to be computed:

$$c = [c_1 c_2 \ldots c_K]^T \tag{8}$$

$$s = [s_1 s_2 \ldots s_K]^T \tag{9}$$

The solution to the least-squares problem (3) is then given by the normal equations [6]:

$$(B^T W^T W B) \begin{bmatrix} c \\ s \end{bmatrix} = B^T W^T W x \tag{10}$$

where $W = diag\,(w(-N), w(-N+1), \ldots, w(N))$ is a diagonal matrix with the symmetric window w for diagonal and x is a $(2N+1) - by - 1$ vector that holds the original signal $x = [x(-N)x(-N+1)\ldots x(N)]^T$. Note that $(B^T W^T W B)^{-1} B^T W^T$ is the pseudoinverse matrix that projects the (weighted) signal into the subspace of the weighted harmonic sines and cosines.

Using simple trigonometric algebra and the fact that the window w is a symmetric one, we have that

$$\begin{aligned} B^T W^T W B &= \begin{bmatrix} C^T W^T W C & C^T W^T W S \\ S^T W^T W C & S^T W^T W S \end{bmatrix} \\ &= \begin{bmatrix} A_c & \mathbf{0} \\ \mathbf{0} & A_s \end{bmatrix} \end{aligned} \tag{11}$$

and

$$B^T W^T W x = \begin{bmatrix} C^T W^T W x \\ S^T W^T W x \end{bmatrix} = \begin{bmatrix} b_c \\ b_s \end{bmatrix} \tag{12}$$

where A_c and A_s are the $K - by - K$ cosine and sine, respectively, weighted correlation matrices, b_c and b_s are the cosine and sine $K - by - 1$ projection vectors, and $\mathbf{0}$ is the $K - by - K$ zero matrix. Therefore, from (10) we get the following two systems to solve:

$$A_c c = b_c \tag{13}$$

$$A_s s = b_s \tag{14}$$

It can be shown, that the correlation matrices A_c and A_s are symmetric Toeplitz matrices, so the two systems can be solved efficiently by Levinson-type algorithms [1], [2], [3]. We propose to solve (13) and (14) by encoding the inverse matrices A_c^{-1} and A_s^{-1}.

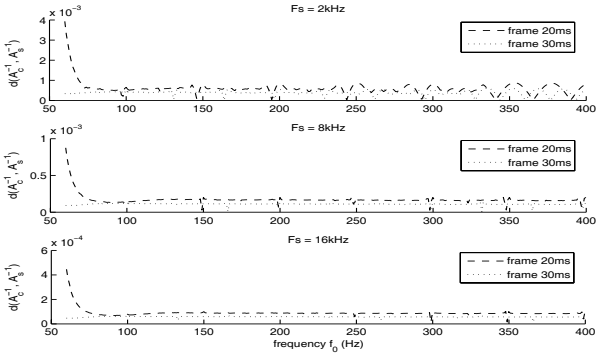

Fig. 1. Similarity measurement using Hamming window.

3. FAST HARMONIC ANALYSIS USING ENCODED MATRICES

This section presents the encoding and decoding algorithm for the inverse matrices A_c^{-1} and A_s^{-1}. Initially, it will be shown that $A_c^{-1} \approx A_s^{-1}$ for many interesting frame durations (20ms and 30ms), sampling rates F_s = { 2 kHz, 8 kHz, 16 kHz } and for all integer fundamental frequencies f_0 between 60 Hz and 400 Hz. Then we will define $A \equiv A_c^{-1}$ and present an encoding algorithm $Q(\cdot)$ that encodes A to $\hat{A} = Q^{-1}(Q(A))$. Finally, the quality of the approximation made by using the encoded matrix $\hat{A}$, for the cases under examination, will be addressed.

3.1. Similarity between A_c^{-1} and A_s^{-1}

The similarity between the inverse matrices A_c^{-1} and A_s^{-1} was measured for all examined cases of sampling rates, frame durations and ω_0, using the *Hamming* window. As a distance measure we used the element-wise mean value of the absolute difference of the product of the first inverse matrix and the second matrix, with the identity matrix I:

$$d(A_1^{-1}, A_2^{-1}) = mean \| A_1^{-1} A_2 - I \| \tag{15}$$

The results of the two measurements: $d(A_c^{-1}, A_s^{-1})$ and $d(A_s^{-1}, A_c^{-1})$ are very close, so we chose to present only the former for clarity.

As shown in Figure 1, the distance $d(A_c^{-1}, A_s^{-1})$ is well below 10^{-3} for the majority of the examined cases. Note that this corresponds to a mean relative error of 0.1% because the elements of the product $A_c^{-1} A_s$ should approximate the identity matrix I. Similar results were obtained for other commonly used windows like *Hanning*, *Rectangular*, and *Blackman*. Therefore, for the rest of the paper, the matrix $A \equiv A_c^{-1} \approx A_s^{-1}$ will serve as an approximation of the inverse of both matrices A_c and A_s.

3.2. Encoding Algorithm

It is unrealistic to store one inverse matrix for every possible f_0. The storage requirements for A can be reduced if we can exploit the structure of the matrix. The elements of A have a specific pattern of similarity in the columns of A. Each column is similar to the other columns in accordance to a shift in row sense. In other words, A is similar to a Toeplitz matrix. Note that the inverse of a symmetric Toeplitz matrix may not be a Toeplitz matrix. Additionally, the main pattern is symmetric. We developed an encoding algorithm that selects representative patterns from the columns of A. These representative patterns are used to create the decoded matrix $\hat{A}$.

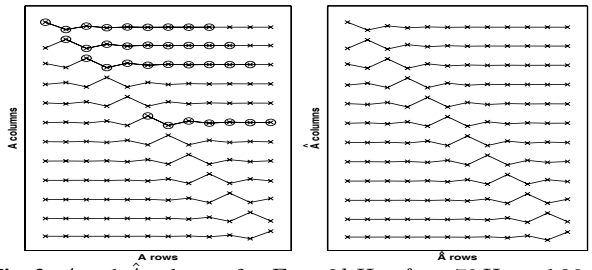

Fig. 2. A and $\hat{A}$ columns for $F_s = 2kHz$, $f_0 = 70Hz$ and $20ms$ frame. Circled are the selected representative patterns.

The selection of the representative patterns is made using an energy criterion so that the decoded columns retain 99.999% of the energy of the original columns. Since the pattern of the center column is approximately symmetric, only half of it needs to be stored. A variable number of patterns (1 to 4) may be extracted for each matrix A. The patterns are extracted from columns 1,2,3 and $\lfloor K/2 \rfloor$. The main representative pattern, p_0, is created from the $\lfloor K/2 \rfloor$-th (center) column and it is always kept. Let p_1, p_2 and p_3 be the representative patterns from the 1^{st}, the 2^{nd} and the 3^{rd} column, respectively. From these three patterns, only the ones that represent columns for which p_0 does not satisfy the energy criterion are kept. Finally, all kept representative patterns are extended to include as many elements as the longest one, and may be zero-padded if they do not have enough elements. The encoded representation of A consists of these patterns and requires only a few parameters per matrix.

An example of a matrix A and its compressed version $\hat{A}$ is shown in Figure 2. The figure plots the columns of A and the columns of the corresponding matrix $\hat{A}$. The representative patterns in the leftmost matrix A are circled. The rightmost matrix $\hat{A}$ is constructed using only the circled representative patterns. It is evident that $\hat{A}$ captures the coarse structure of A. The number of parameters required to encode *all matrices* $A(f_0)$ for $f_0 = 60, 61, ..., 400Hz$ and the corresponding compression ratios are shown in Table 1. Clearly, the memory requirements for the representative patterns p_i,$i = 0, ..., 3$ are quite low.

$F_s(kHz)$	Frame(ms)	Parameters	Compression Ratio
2	20	1387	7.2
2	30	506	19.8
8	20	1833	103.2
8	30	504	375.4
16	20	2069	392.8
16	30	503	1616.0

Table 1. Number of parameters needed to encode all matrices for $f_0 = 60, 61, ..., 400Hz$. A Hamming window was used.

The decoded matrix $\hat{A}$ is decoded from the stored representative patterns. The center column representative pattern is mirrored and concatenated to itself, to approximate the original center column and is copied to all columns using the appropriate shift. The representative patterns of the first, second and third column - whichever are kept- are copied to their respective column, with the appropriate mirroring and concatenation for the second and third pattern. These are also flipped and copied to the last, second to last and third to last column accordingly.

The representative patterns p_i evolve smoothly with respect to f_0. Therefore, we can interpolate the representative patterns

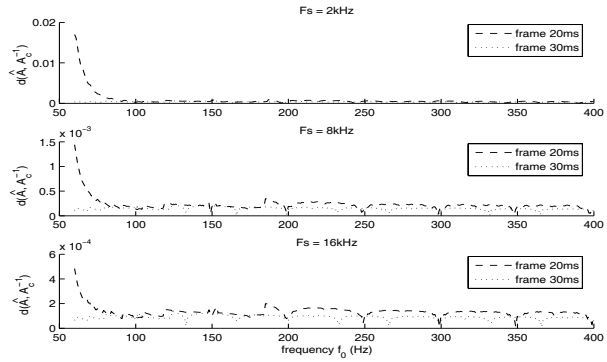

Fig. 3. Encoding quality measurement using Hamming window.

for values of f_0 that were not used at the encoding stage. For example, the patterns p_i for $f_0 = 100.5Hz$ can be taken from the linear interpolation of the nearest patterns at $f_0 = 100Hz$ and $f_0 = 101Hz$: $p_i(100.5) = 0.5(p_i(100) + p_i(101))$.

The pattern structure of A is broken when the harmonic analysis includes sinusoids that are near the Nyquist frequency, $F_s/2$. Therefore, all the experiments in the paper were conducted with a maximum harmonic frequency that is slightly lower than $F_s/2$. In particular, the cutoff frequencies for $F_s = 2kHz$, $F_s = 8kHz$ and $F_s = 16kHz$ were $0.9kHz$, $3.7kHz$ and $7.6kHz$, respectively.

The complexity of the decoding process depends on the number of harmonics K and the size of the patterns. Since the size of patterns is bounded to a few coefficients, the complexity of the proposed algorithm is linear, i.e. $O(n)$. This is a significant improvement, compared to the best Levinson solvers [3], [4] which require a complexity of $O(n^2)$. In fact, the decoding process is just a multiplication of b_c (or b_s) with a sparse matrix generated by the representative patterns.

3.3. Quality of the Approximation

The approximation made by the proposed encoding algorithm is evaluated with the distance measure in (15). The distances $d(\hat{A}, A_c^{-1})$ and $d(\hat{A}, A_s^{-1})$ were measured for $f_0 = 60, ..., 400Hz$ and several sampling rates, windows, frame durations. However, the corresponding distances $d(\hat{A}, A_c^{-1})$ and $d(\hat{A}, A_s^{-1})$ are very close in a numerical sense, therefore for clarity, we will present results regarding only $d(\hat{A}, A_c^{-1})$. The results are depicted in Figure 3 where it is shown that distance $d(\hat{A}, A_c^{-1})$ is below 0.02 for the case of $F_s = 2kHz$ and well below 0.01 for the rest of the examined cases. Note that the measurements were made using a Hamming window and that similar results were obtained for other windows.

4. FAST HARMONIC SYNTHESIS

The computation of the cosine and sine bases, used in (13) and (14), requires a considerable portion of the complexity of the HM analysis. However, the cosine and sine functions can be computed iteratively over time by [5]:

$$\cos(k\omega_o(n+1)) \simeq (1-\alpha)\cos(k\omega_o n) - \beta \sin(k\omega_o n)$$
$$\sin(k\omega_o(n+1)) \simeq (1-\alpha)\sin(k\omega_o n) + \beta \cos(k\omega_o n)$$

where $\alpha = 2\sin^2(0.5k\omega_o)$ and $\beta = \sin(k\omega_o)$. These computations need 2 MAC (Multiply-Accumulate) operations for each cosine or sine evaluation. The complexity can be further reduced

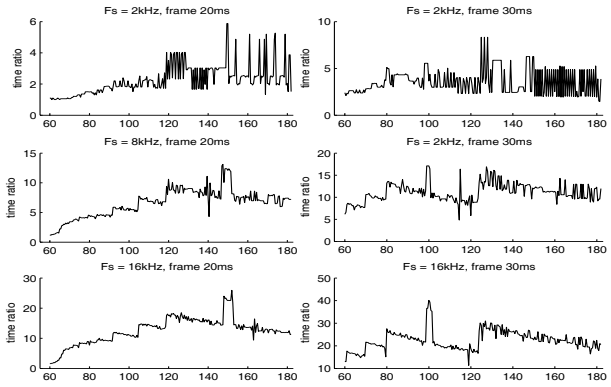

Fig. 4. Speed comparison using Hamming window for the frequency range of [60:0.5:182]Hz.

if the recurrence is taken over the harmonic frequencies:

$$\cos(k\omega_o n) = 2\cos((k-1)\omega_o n)\cos(\omega_o n) - \cos((k-2)\omega_o n)$$
$$\sin(k\omega_o n) = 2\sin((k-1)\omega_o n)\cos(\omega_o n) - \sin((k-2)\omega_o n)$$

With proper implementation, these computations need only 1 MAC operation each. Furthermore, the symmetry and antisymmetry of C and S over the rows can be exploited to slice the cost for the computation of the corresponding matrices. Note that the central row of C and S have constant values of 1 and 0, respectively. We use the following combination of the presented recurrence relations. Initially, we use the recurrence over the rows of C, S (time index n) to compute the sine/cosines of the first harmonic. Then we use the recurrence over the frequencies to compute the rest of the harmonics.

5. EXPERIMENTS

A speed comparison was made between the proposed algorithm and the Levinson algorithm [1]. The Levinson algorithm was chosen because it is a well known and widely implemented algorithm; therefore a suitable point of reference. The motivation behind the comparison is to provide a reference example of the behavior of the proposed algorithm in specific conditions.

A white noise signal was analyzed for fundamental frequencies f_0 in the range of 60 Hz to 400 Hz with step 0.5 Hz at the sampling rates and frame sizes under examination. Both systems in (13), (14) were solved with the proposed algorithm and the Levinson algorithm [1]. Six different experiments were conducted, one for each combination of the three sampling rates $F_s = \{2\,kHz$, $8\,kHz$, $16\,kHz\}$ and the 2 window sizes $\{$ 20 ms, 30 ms $\}$. For each comparison, the algorithms were repeated 100.000 times to reduce the effect of the operating system on the measurements. The experiments were conducted using a 3 GHz Pentium 4 PC and optimized ANSI-C implementations. Figure 4 shows the ratio between the execution time of the Toeplitz solver and the execution time of the proposed algorithm. For example, a ratio of 5 means that the proposed algorithm is 5 times faster than the Toeplitz solver. The proposed algorithm is 2-30 times faster than the Levinsion algorithm. The gain increases as the number of harmonics K increases (or equivalently the F_s) or the size of the window increases. A larger window reduces the length of the representative patterns and this is clearly reflected into the speed measurements. Furthermore, the speed measurements indicate that the proposed algorithm has linear complexity $O(n)$. Note that the fluctuations in the time ratio are due to the fact that for the non-integer fundamental frequencies our algorithm performs an interpolation between the representative patterns of the two nearest integer fundamental frequencies.

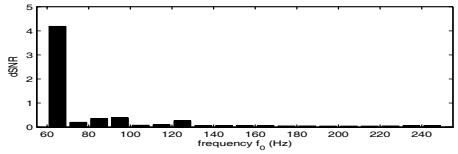

Fig. 5. Average segmented SNR degradation for several f_0 intervals.

The performance of the proposed algorithm was also evaluated in terms of segmental SNR. Tests of analysis/synthesis of narrowband speech signals (F_s = 4 kHz) using 20ms frames weighted by a Hanning window were conducted. Estimation of pitch was used for the voiced frames, while for the unvoiced frames, a constant fundamental frequency $f_0 = 100\ Hz$ was used. The comparison was made using 512 narrowband utterances (256 males, 256 females). Let SNR_{Lev} be the segmental SNR provided by the Levinson algorithm and SNR_{fast} be the segmental SNR provided by the proposed algorithm. The difference $dSNR = SNR_{Lev} - SNR_{fast}$ was taken on a frame-by-frame basis. Figure 5, depicts the average $dSNR$ for several f_0 intervals. Note that the SNR degradation is negligible for most frequencies. However, in lower frequencies ($f_0 < 70$ Hz) the degradation is more evident. That is to be expected, since as we showed in figures 1 and 3 our algorithm doesn't perform well for this range. This is not a significant problem because such pitch values are very rare and the SNR is already very high due to the dense frequency sampling. In order to reduce the SNR loss, the number of representative patterns must be increased.

6. CONCLUSION

An $O(n)$ algorithm for Harmonic Model (HM) analysis was presented. The algorithm is based on a lookup table of encoded matrices and provides an approximate solution to the symmetric Toeplitz systems that appear in HM analysis. Experiments indicate that there are considerable speed improvements (2-30 times) over the Levinson algorithm for many common analysis cases encountered in speech processing. Furthermore, the degradation in terms of SNR is minimal.

7. REFERENCES

[1] N. Levinson, "The Wiener RMS (root mean square) error criterion in filter design and prediction," *Journal Math. Phys.*, vol. 25, pp. 261–278, 1947.

[2] P Delsarte and Y. Genin, "An extension of the split Levinson algorithm and its relatives to the joint process estimation problem," *IEEE Trans. Inform. Theory*, vol. 34, pp. 482–485, 1989.

[3] A. Melman, "The even-odd split Levinson algorithm for Toeplitz systems," *SIAM Journal Matrix Analysis and Applications.*, vol. 23, 2001.

[4] G Heinig and K Rost, "Split algorithms for symmetric toeplitz matrices with arbitrary rank profile," *Numerical Linear Algebra with Applications*, vol. 12, no. 2-3, pp. 141–151, 2004.

[5] Yannis Stylianou, "A simple and fast way for generating a harmonic signal," *IEEE Signal Processing Letters*, vol. 7, no. 5, May 2000.

[6] C.L. Lawson and R.J. Hanson, *Solving Least-Squares Problems*, Prentice Hall, 1974.