IMPLEMENTATION AND EVALUATION OF THE VANDERMONDE TRANSFORM

Tom Bäckström, Johannes Fischer* and Daniel Boley†*

*International Audio Laboratories Erlangen, Friedrich-Alexander University Erlangen-Nürnberg (FAU)
Am Wolfsmantel 33, 91058 Erlangen, Germany, tom.backstrom@audiolabs-erlangen.de
† Deparment of Computer Science&Engineering, University of Minnesota, USA.

ABSTRACT

The Vandermonde transform was recently presented as a time-frequency transform which, in difference to the discrete Fourier transform, also decorrelates the signal. Although the approximate or asymptotic decorrelation provided by Fourier is sufficient in many cases, its performance is inadequate in applications which employ short windows. The Vandermonde transform will therefore be useful in speech and audio processing applications, which have to use short analysis windows because the input signal varies rapidly over time. Such applications are often used on mobile devices with limited computational capacity, whereby efficient computations are of paramount importance.

Implementation of the Vandermonde transform has, however, turned out to be a considerable effort: it requires advanced numerical tools whose performance is optimized for complexity and accuracy. This contribution provides a baseline solution to this task including a performance evaluation.

Index Terms— time-frequency transforms, decorrelation, Vandermonde matrix, Toeplitz matrix, warped discrete Fourier transform

1. INTRODUCTION

The discrete Fourier transform is one of the most fundamental tools in digital signal processing. It provides a physically motivated representation of an input signal in the form of frequency components. Since the Fast Fourier Transform (FFT) calculates the discrete Fourier transform also with very low computational complexity $\mathcal{O}(N \log N)$, it has become one of the most important tools of digital signal processing [1].

Although celebrated, the discrete Fourier transform has a blemish: It does not decorrelate signal components completely (for a numerical example, see Section 4). Only when the transform length converges to infinity do the components become orthogonal [2]. Such approximate decorrelation is in many applications good enough. However, applications which employ relatively small transforms such as many speech and audio processing algorithms, the accuracy of this approximation limits the overall efficiency of algorithms. For example, the speech coding standard AMR-WB [3] employs windows of length $N = 64$. Practice has shown that performance of the discrete Fourier transform is in this case insufficient and consequently, most mainstream speech codecs use time-domain encoding [3].

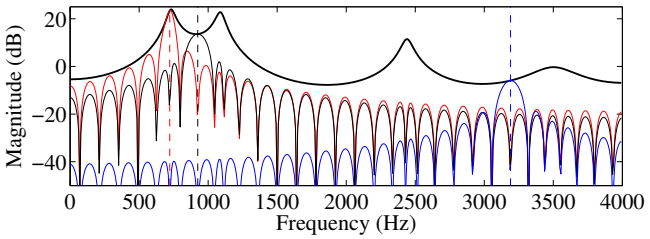

Fig. 1. Characteristics of a Vandermonde transform; the thick black line illustrates the (non-warped) Fourier spectrum of a signal and the red, black and blue lines are the response of pass-band filters of three selected frequencies, filtered with the input signal. The Vandermonde factorization size is 64.

There are naturally plenty of transforms which provide decorrelation of the input signal, such as the Karhunen-Loève transform (KLT) [4]. However, the components of the KLT are abstract entities without a physical interpretation as simple as the Fourier transform. A physically motivated domain, on the other hand, allows straightforward implementation of physically motivated criteria into the processing methods. A transform which provides both a physical interpretation and decorrelation is therefore desired.

We have recently presented a transform, called the Vandermonde transform, which has both of the preferred characteristics [5]. It is based on a decomposition of a Hermitian Toeplitz matrix into a product of a diagonal matrix and a Vandermonde matrix. This factorization is actually also known as the Carathéodory parametrization of covariance matrices [6] and is very similar to the Vandermonde factorization of Hankel matrices [7].

For the special case of positive definite Hermitian Toeplitz matrices, the Vandermonde factorization will correspond to a frequency-warped discrete Fourier transform. In other words, it is a time-frequency transform which provides signal components sampled at frequencies which are not necessarily uniformly distributed. The Vandermonde transform thus pro-

vides both the desired properties: decorrelation and a physical interpretation.

While the existence and properties of the Vandermonde transform have been analytically demonstrated, the purpose of the current work is, firstly, to collect and document existing practical algorithms for Vandermonde transforms. These methods have appeared in very different fields, including numerical algebra, numerical analysis, systems identification, time-frequency analysis and signal processing, whereby they are often hard to find. This paper is thus a review of methods which provide a joint platform for analysis and discussion of results. Secondly, we provide numerical examples as a baseline for further evaluation of the performance of the different methods.

2. VANDERMONDE TRANSFORM

This section provides a brief introduction to Vandermonde transforms. For a more comprehensive motivation and discussion about applications, we refer to [5].

A Vandermonde matrix $\mathbf{V}$ is defined by the scalars ν_k as

$$\mathbf{V} = \begin{bmatrix} 1 & \nu_0 & \nu_0^2 & \dots & \nu_0^{N-1} \\ 1 & \nu_1 & \nu_1^2 & \dots & \nu_1^{N-1} \\ \vdots & \vdots & & \vdots & \\ 1 & \nu_{N-1} & \nu_{N-1}^2 & \dots & \nu_{N-1}^{N-1} \end{bmatrix}. \quad (1)$$

It is full rank if scalars ν_k are distinct ($\nu_k \neq \nu_h$ for $k \neq h$) and its inverse has an explicit formula [8].

A symmetric Toeplitz matrix $\mathbf{T}$ is defined by scalars τ_k as

$$\mathbf{T} = \begin{bmatrix} \tau_0 & \tau_1 & \dots & \tau_{N-1} \\ \tau_1 & \tau_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \tau_1 \\ \tau_{N-1} & \dots & \tau_1 & \tau_0 \end{bmatrix}. \quad (2)$$

If $\mathbf{T}$ is positive definite, then it can be factorized as [5]

$$\mathbf{T} = \mathbf{V}^* \mathbf{\Lambda} \mathbf{V}, \quad (3)$$

where $\mathbf{\Lambda}$ is a diagonal matrix with real and strictly positive entries $\lambda_{kk} > 0$ and the exponential series $\mathbf{V}$ are all on the unit circle $\nu_k = \exp(i\beta_k)$. This form is also known as the Carathéodory parametrization of a Toeplitz matrix [6].

We present here two uses for the Vandermonde transform: either as a decorrelating transform or as a replacement for a convolution matrix. Consider first a signal $\mathbf{x}$ which has the autocorrelation matrix $E[\mathbf{x}\mathbf{x}^*] = \mathbf{R}_x$. Since the autocorrelation matrix is positive definite, symmetric and Toeplitz, we can factorize it as $\mathbf{R} = \mathbf{V}^* \mathbf{\Lambda} \mathbf{V}$. It follows that if we apply the transform

$$\mathbf{y}_d = \mathbf{V}^{-*} \mathbf{x} \quad (4)$$

where $\mathbf{V}^{-*}$ is the inverse Hermitian of $\mathbf{V}$, then the autocorrelation matrix of $\mathbf{y}_d$ is

$$\begin{aligned} \mathbf{R}_y &= E[\mathbf{y}_d \mathbf{y}_d^*] = \mathbf{V}^{-*} E[\mathbf{x}\mathbf{x}^*] \mathbf{V}^{-1} = \mathbf{V}^{-*} \mathbf{R}_x \mathbf{V}^{-1} \\ &= \mathbf{V}^{-*} \mathbf{V}^* \mathbf{\Lambda} \mathbf{V} \mathbf{V}^{-1} = \mathbf{\Lambda}. \end{aligned} \quad (5)$$

The transformed signal $\mathbf{y}_d$ is thus uncorrelated. The inverse transform is

$$\mathbf{x} = \mathbf{V}^* \mathbf{y}_d. \quad (6)$$

As a heuristic description, we can say that the forward transform $\mathbf{V}^{-*}$ contains in its kth row a filter whose pass-band is at frequency $-\beta_k$ and the stop-band output for $\mathbf{x}$ has low energy. Specifically, the spectral shape of the output is close to that of an AR-filter with a single pole on the unit circle. Note that since this filterbank is signal adaptive, we consider here the output of the filter rather than the frequency response of the basis functions.

The backward transform $\mathbf{V}^*$ in turn has exponential series in its columns, such that $\mathbf{x}$ is a weighted sum of the exponential series. In other words, the transform is a warped time-frequency transform. Fig. 1 demonstrates the discrete (non-warped) Fourier spectrum of an input signal $\mathbf{x}$ and frequency responses of selected rows of $\mathbf{V}^{-*}$.

The Vandermonde transform for evaluation of a signal in a convoluted domain can be constructed as follows. Let $\mathbf{C}$ be a convolution matrix and $\mathbf{x}$ the input signal. Consider the case where our objective is to evaluate the convoluted signal $\mathbf{y}_c = \mathbf{C}\mathbf{x}$. Such evaluation appears, for example, in speech codecs employing ACELP, where quantization error energy is evaluated in a perceptual domain and where the mapping to the perceptual domain is described by a filter [3].

The energy of $\mathbf{y}_c$ is

$$\begin{aligned} \|\mathbf{y}_c\|^2 &= \|\mathbf{C}\mathbf{x}\|^2 = \mathbf{x}^* \mathbf{C}^* \mathbf{C} \mathbf{x} = \mathbf{x}^* \mathbf{R}_c \mathbf{x} \\ &= \mathbf{x}^* \mathbf{V}^* \mathbf{\Lambda} \mathbf{V} \mathbf{x} = \|\mathbf{\Lambda}^{1/2} \mathbf{V} \mathbf{x}\|^2. \end{aligned} \quad (7)$$

The energy of $\mathbf{y}_c$ is thus equal to the energy of the transformed and scaled signal

$$\mathbf{y}_v = \mathbf{\Lambda}^{1/2} \mathbf{V} \mathbf{x}. \quad (8)$$

We can thus equivalently evaluate signal energy in the convolved or the transformed domain, $\|\mathbf{y}_c\|^2 = \|\mathbf{y}_v\|^2$. The inverse transform is obviously

$$\mathbf{x} = \mathbf{V}^{-1} \mathbf{\Lambda}^{-1/2} \mathbf{y}_v. \quad (9)$$

The forward transform $\mathbf{V}$ has exponential series in its rows, whereby it is a warped Fourier transform. Its inverse $\mathbf{V}^{-1}$ has filters in its columns, with pass-bands at β_k. In this form the frequency response of the filter-bank is equal to a discrete Fourier transform. It is only the inverse transform which employs what is usually seen as aliasing components in order to enable perfect reconstruction.

3. PRACTICAL ALGORITHMS

For using Vandermonde transforms, we need effective algorithms for determining as well as applying the transforms. In this section we will discuss available algorithms. Let us begin with application of transforms since it is the more straightforward task.

Multiplications with $\mathbf{V}$ and $\mathbf{V}^*$ are straightforward and can be implemented in $\mathcal{O}(N^2)$. To reduce the storage requirements, we show here algorithms where exponents ν_k^h need not be explicitly evaluated for $h > 1$. Namely, if $\mathbf{y} = \mathbf{V}\mathbf{x}$ and the elements of $\mathbf{x}$ are ξ_k, then the elements η_k of $\mathbf{y}$ can be determined with the recurrence

$$\begin{cases} \tau_{h,0} = \xi_{N-1} \\ \tau_{h,k} = \xi_{N-1-k} + \nu_h \tau_{h,k-1}, \quad \text{for } 1 \leq k < N \\ \eta_h \quad = \tau_{h,N-1}. \end{cases} \tag{10}$$

Here $\tau_{h,k}$ is a temporary scalar, of which only the current value needs to be stored. The overall recurrence has N steps for N components, whereby overall complexity is $\mathcal{O}(N^2)$ and storage constant. A similar algorithm can be readily written for $\mathbf{y} = \mathbf{V}^*\mathbf{x}$.

Multiplication with the inverse Vandermonde matrices $\mathbf{V}^{-1}$ and $\mathbf{V}^{-*}$ is a slightly more complex task but fortunately relatively efficient methods are already available from literature [9, 10]. The algorithms are simple to implement and for both $\mathbf{x} = \mathbf{V}^{-1}\mathbf{y}$ and $\mathbf{x} = \mathbf{V}^{-*}\mathbf{y}$ the complexity is $\mathcal{O}(N^2)$ and storage linear $\mathcal{O}(N)$. However, the algorithm includes a division at every step, which has in many architectures a high constant cost.

Although the above algorithms for multiplication by the inverses are exact in an analytic sense, practical implementations are numerically unstable for large N. In our experience, computations with matrices up to a size of $N \sim 64$ is sometimes possible, but beyond that the numerical instability renders these algorithms useless as such. A practical solution is Leja-ordering of the roots ν_k [11] which is equivalent to Gaussian Elimination with Partial Pivoting [12]. The main idea behind Leja-ordering is to reorder the roots in such a way that the distance of a root ν_k to its predecessors $0 \ldots (k-1)$ is maximized. By such reordering the denominators appearing in the algorithm are maximized and values of intermediate variables are minimized, whereby the contributions of truncation errors are also minimized. Implementation of Leja-ordering is simple and can be achieved with complexity $\mathcal{O}(N^2)$ and storage $\mathcal{O}(N)$.

The final hurdle is then obtaining the factorization, that is, the roots ν_k and when needed, the diagonal values λ_{kk}. From [5] we know that the roots can be obtained by solving

$$\mathbf{R}\mathbf{a} = [1\,1 \ldots 1]^T, \tag{11}$$

where $\mathbf{a}$ has elements α_k. Then $\nu_0 = 1$ and the remaining roots $\nu_1 \ldots \nu_N$ are the roots of polynomial $A(z) = \sum_{k=0}^{N-1} \alpha_k z^{-k}$. We can readily show that this is equivalent with solving the Hankel system

$$\begin{bmatrix} \tau_{N-1} & \cdots & \tau_1 & \tau_0 \\ \vdots & \iddots & \tau_0 & \tau_1 \\ \tau_1 & \iddots & \iddots & \vdots \\ \tau_0 & \tau_1 & \ldots & \tau_{N-1} \end{bmatrix} \begin{bmatrix} \hat{\alpha}_1 \\ \hat{\alpha}_2 \\ \vdots \\ \hat{\alpha}_N \end{bmatrix} = - \begin{bmatrix} \tau_1 \\ \tau_2 \\ \vdots \\ \tau_N \end{bmatrix} \tag{12}$$

where $\tau_N = -\frac{1}{\alpha_0} \sum_{k=1}^{N-1} \alpha_{k+1} \tau_{N-k}$. The roots ν_k are then the roots of $\hat{A}(z) = 1 + \sum_{k=1}^{N} \hat{\alpha}_k z^{-k}$.

Since factorization of the original Toeplitz system Eq. 11 is equivalent with Eq. 12, we can use a fast algorithm for factorization of Hankel matrices [13]. This algorithm returns a tridiagonal matrix whose eigenvalues correspond to the roots of $\hat{A}(z)$. The eigenvalues can then be obtained in $\mathcal{O}(N^2)$ by applying the LR algorithm, or in $\mathcal{O}(N^3)$ by the standard non-symmetric QR-algorithm [14, 15]. The roots obtained this way are approximations, whereby they might be slightly off the unit circle. It is then useful to normalize the absolute value of the roots to unity, and refine with 2 or 3 iterations of Newton's method. The complete process has a computational cost of $\mathcal{O}(N^2)$.

The last step in factorization is to obtain the diagonal values $\mathbf{\Lambda}$. Observe that

$$\mathbf{R}\mathbf{e} = \mathbf{V}^*\mathbf{\Lambda}\mathbf{V}\mathbf{e} = \mathbf{V}^*\boldsymbol{\lambda} \tag{13}$$

where $\mathbf{e} = [1\,0 \ldots 0]^T$ and $\boldsymbol{\lambda}$ is a vector containing the diagonal values of $\mathbf{\Lambda}$. In other words, by calculating

$$\boldsymbol{\lambda} = \mathbf{V}^{-*}(\mathbf{R}\mathbf{e}), \tag{14}$$

we obtain the diagonal values λ_{kk}. This inverse can be calculated with the methods discussed above, whereby the diagonal values are obtained with complexity $\mathcal{O}(N^2)$.

In summary, the steps required for factorization of a matrix $\mathbf{R}$ are

1. Solve Eq. 11 for $\mathbf{a}$ using Levinson-Durbin or other classical methods.
2. Extend autocorrelation sequence by $\tau_N = -\frac{1}{\alpha_0} \sum_{k=1}^{N-1} \alpha_{k+1} \tau_{N-k}$.
3. Apply tridiagonalization algorithm of [13] on sequence τ_k.
4. Solve eigenvalues ν_k using either the LR- or the symmetric QR-algorithm [14, 15].
5. Refine root locations by scaling ν_k to unity and a few iterations of Newton's method.
6. Determine diagonal values λ_{kk} using Eq. 14.

4. EXPERIMENTS

Let us begin with a numerical example that demonstrates the concepts used. Here matrix $\mathbf{C}$ is a convolution matrix corresponding to the trivial filter $1 + z^{-1}$, matrix $\mathbf{R}$ its autocorrelation, matrix $\mathbf{V}$ the corresponding Vandermonde matrix obtained with the algorithm in Section 3, matrix $\mathbf{F}$ is

the discrete Fourier transform matrix and the matrices $\mathbf{\Lambda}_V$ and $\mathbf{\Lambda}_F$ demonstrate the diagonalization accuracy of the two transforms. We can thus define

$$\mathbf{C} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \quad \mathbf{R} = \mathbf{C}\mathbf{C}^* = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix},$$
$$\mathbf{V} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & i & -1 \\ 1 & -i & -1 \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & e^{-\frac{i\pi 2}{3}} & e^{+\frac{i\pi 2}{3}} \\ 1 & e^{+\frac{i\pi 2}{3}} & e^{-\frac{i\pi 2}{3}} \end{bmatrix}, \tag{15}$$

whereby we can evaluate the diagonalization with

$$\mathbf{\Lambda}_V = |\mathbf{V}^{-*}\mathbf{R}\mathbf{V}^{-1}| = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix},$$
$$\mathbf{\Lambda}_F = |\mathbf{F}^{-*}\mathbf{R}\mathbf{F}^{-1}| = \begin{bmatrix} 1.11 & 0.111 & 0.111 \\ 0.111 & 0.444 & 0.222 \\ 0.111 & 0.222 & 0.444 \end{bmatrix}. \tag{16}$$

We can here see that with the Vandermonde transform we obtain a perfectly diagonal matrix $\mathbf{\Lambda}_V$. The performance of the discrete Fourier transform is far from optimal, since the off-diagonal values are clearly non-zero. As a measure of performance, we can calculate the ratio of the absolute sums of off- and on-diagonal values, which is zero for the Vandermonde factorization and 0.444 for the Fourier transform.

We can then proceed to evaluate the implementations described in Section 3. We have implemented each algorithm in MATLAB with the purpose of providing a performance baseline upon which future works can compare and to find eventual performance bottlenecks. We will consider performance in terms of complexity and accuracy.

To determine the performance of the factorization, we will compare the Vandermonde factorization to the discrete Fourier and Karhunen-Loève transforms, the latter applied with the eigenvalue decomposition. We have applied the Vandermonde factorization using two methods, firstly, the algorithm described in this article (V_1), and secondly, the approach described in [5] using the built-in root-finding function provided by MATLAB (V_2). Since this MATLAB function is a finely tuned generic algorithm, we would expect to obtain accurate results but with higher complexity than our purpose-built algorithm.

As data for all our experiments we used the set of speech, audio and mixed sound samples used in evaluation of the MPEG USAC standard [16] with a sampling rate of 12.8 kHz. The audio samples were windowed with Hamming windows to the desired length and their autocorrelations were calculated. To make sure the autocorrelation matrices are positive definite, the main diagonal was multiplied with $(1 + 10^{-5})$.

For performance measures we used computational complexity in terms of normalized running time and accuracy in terms of how close $\hat{\mathbf{\Lambda}} = \mathbf{V}^{-*}\mathbf{R}\mathbf{V}^{-1}$ is to a diagonal matrix, measured by the ratio of absolute sums of off- and on-diagonal elements. Results are listed in Tables 1 and 2.

Table 1. Complexity of factorization algorithms for different window lengths N in terms of normalized running time.

N	16	32	64	128	256	512
V_1	1.00	3.02	10.13	35.96	131.80	496.91
V_2	1.00	2.10	8.77	90.61	634.17	4056.62
KLT	1.00	4.33	8.93	30.59	109.53	419.76

Table 2. Accuracy of factorization algorithms for different window lengths N in terms of $\log_{10}$ of ratio of absolute sums of off- and on-diagonal values of $\hat{\mathbf{\Lambda}} = \mathbf{V}^{-*}\mathbf{R}\mathbf{V}^{-1}$.

N	16	32	64	128	256	512
FFT	-0.22	-0.16	-0.13	-0.11	-0.08	-0.07
V_1	-2.36	-2.14	-1.93	-1.72	-1.26	-0.97
V_2	-13.99	-13.56	-13.11	-12.67	-12.14	-11.56
KLT	-14.56	-14.24	-14.07	-13.89	-13.65	-13.23

Note that here it is not sensible to compare the running times between algorithms, only the increase in complexity as a function of frame size, because the built-in MATLAB functions have been implemented in a different language than our own algorithms. We can see that the complexity of the proposed algorithm V_1 increases with a comparable rate as the KLT, while the algorithm employing root-finding functions of MATLAB V_2 increases more. The accuracy of the proposed factorization algorithm V_1 is not yet optimal. However, since the root-finding function of MATLAB V_2 yields comparable accuracy as the KLT, we conclude that improvements are possible by algorithmic improvements.

The second experiment is application of transforms to determine accuracy and complexity. Firstly, we apply Eqs. 4 and 9, whose complexities are listed in Table 3. Here we can see that matrix multiplication of KLT and the built-in solution of matrix systems of MATLAB V_2 have roughly the same rate of increase in complexity, while the proposed methods for Eqs. 4 and 9 have a much smaller increase. The FFT is naturally faster than all the other approaches.

Finally, to obtain the accuracy of Vandermonde solutions, we apply the forward and backward transforms in sequence. The Euclidean distances between original and reconstructed vectors are listed in Table 4. We can observe, firstly, that the FFT and KLT algorithms are, as expected, the most accurate, since they are based on orthonormal transforms. Secondly we can see that the accuracy of the proposed algorithm V_1 is slightly lower than the built-in solution of MATLAB V_2, but both algorithms provide sufficient accuracy.

5. DISCUSSION

We have presented implementation details of decorrelating time-frequency transforms using Vandermonde factorization with the purpose of reviewing available algorithms as well as providing performance baselines for further development. While the algorithms were in principle available from previous works, it turns out that getting a system to run requires

Table 3. Complexity of Vandermonde solutions for different window lengths N in terms of normalized running time. Here V_1^{-*} and V_1^{-1} signifies solution of Eqs. 4 and 9 with respective proposed algorithms.

N	16	32	64	128	256	512
FFT	1.00	1.13	1.31	1.99	2.96	3.82
V_1^{-*}	1.00	2.00	4.30	10.17	24.52	68.56
V_1^{-1}	1.00	1.99	4.26	10.14	24.64	69.49
V_2	1.00	1.86	7.57	23.16	78.44	284.80
KLT	1.00	1.31	5.37	8.55	46.25	289.30

Table 4. Accuracy of forward and backward transforms as measured by $\log_{10}(\|\mathbf{x} - \hat{\mathbf{x}}\|^2/\|\mathbf{x}\|^2)$, where $\mathbf{x}$ and $\hat{\mathbf{x}}$ are the original and reconstructed vectors.

N	16	32	64	128	256	512
FFT	-15.82	-15.71	-15.66	-15.62	-15.58	-15.55
V_1^{-*}	-14.62	-14.07	-13.43	-12.89	-12.40	-12.11
V_1^{-1}	-15.15	-14.84	-14.51	-14.14	-13.78	-13.42
V_2	-15.38	-15.22	-15.00	-14.80	-14.67	-14.52
KLT	-14.98	-14.85	-14.78	-14.70	-14.61	-14.51

considerable effort. The main challenges are numerical accuracy and computational complexity. The experiments confirm that methods are available with $\mathcal{O}(N^2)$ complexity, although obtaining low complexity simultaneously with numerical stability is a challenge. However, since the generic MATLAB implementations provide accurate solutions, we assert that obtaining high accuracy is possible with further tuning of the implementation.

In conclusion, our experiments show that for Vandermonde solutions, the proposed algorithms have good accuracy and sufficiently low complexity. For factorization, the purpose-built factorization does give better decorrelation than FFT with reasonable complexity, but in accuracy there is room for improvement. The built-in implementations of MATLAB give a satisfactory accuracy, which leads us to the conclusion that accurate $\mathcal{O}(N^2)$ algorithms can be implemented.

6. ACKNOWLEDGEMENTS

Prof. Boley is funded by NSF grant IIS-1319749.

References

[1] A. V. Oppenheim, A. S. Willsky, and S. H. Nawab, *Signals and systems*, Prentice Hall, 1997.

[2] Y. Yemini and J. Pearl, "Asymptotic properties of discrete unitary transforms," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, , no. 4, pp. 366–371, 1979.

[3] W. C. Chu, *Speech coding algorithms: Foundation and evolution of standardized coders*, John Wiley & Sons, 2004.

[4] I. Jolliffe, *Principal component analysis*, Wiley Online Library, 2005.

[5] T. Bäckström, "Vandermonde factorization of Toeplitz matrices and applications in filtering and warping," *IEEE Trans. Signal Process.*, vol. 61, no. 24, pp. 6257–6263, 2013.

[6] P. Stoica and R. L. Moses, *Spectral analysis of signals*, Pearson/Prentice Hall Upper Saddle River, NJ, 2005.

[7] D. L. Boley, F. T. Luk, and D. Vandevoorde, "Vandermonde factorization of a Hankel matrix," *Scientific computing*, pp. 27–39, 1997.

[8] N. Macon and A. Spitzbart, "Inverses of Vandermonde matrices," *The American Mathematical Monthly*, vol. 65, no. 2, pp. 95–100, 1958.

[9] Å. Björck and V. Pereyra, "Solution of Vandermonde systems of equations," *Mathematics of Computation*, vol. 24, no. 112, pp. 893–903, 1970.

[10] G. H. Golub and C. F. van Loan, *Matrix Computations*, John Hopkins University Press, 3rd edition, 1996.

[11] C. F. Pedersen, "Leja ordering LSFs for accurate estimation of predictor coefficients," in *International Conference on Spoken Language Processing (ICSLP)*, Florence, Italy, 2011, ISCA.

[12] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, vol. 48, SIAM, 1996.

[13] D. L. Boley, F. T. Luk, and D. Vandevoorde, "A fast method to diagonalize a Hankel matrix," *Linear algebra and its applications*, vol. 284, no. 1, pp. 41–52, 1998.

[14] D. S. Watkins, *The matrix eigenvalue problem: GR and Krylov subspace methods*, vol. 101, Siam, 2007.

[15] J. H. Wilkinson, *The algebraic eigenvalue problem*, vol. 155, Oxford Univ Press, 1996.

[16] M. Neuendorf, P. Gournay, M. Multrus, J. Lecomte, B. Bessette, R. Geiger, S. Bayer, G. Fuchs, J. Hilpert, N. Rettelbach, R. Salami, G. Schuller, R. Lefebvre, and B. Grill, "Unified speech and audio coding scheme for high quality at low bitrates," in *Acoustics, Speech and Signal Processing. ICASSP 2009. IEEE Int Conf*, 2009, pp. 1–4.