A Low-complexity Algorithm to Search for Legendre Pairs

Sirani M. Perera*

Department of Mathematics, Embry-Riddle Aeronautical University, Daytona Beach, Florida 32114, USA

Ilias S. Kotsireas

Wilfrid Laurier University, Waterloo, Ontario, Canada

Abstract

Legendre pairs constitute an important combinatorial object that can be used to construct Hadamard matrices. In this paper, we search for Legendre pairs exploring matrix structures and obtain a low arithmetic and time complexity algorithm instead of conventional combinatorial algorithms. First, we explore the structure of the Legendre pair matrix equation and study its properties. Next, we study the boundaries of the spectra of the matrices appearing in the Legendre pair matrix equation, using Gershgorin circles. After stating an invariant that characterizes Legendre pairs and their relation to the discrete Fourier transform (DFT) matrix, we propose a low-complexity algorithm, i.e., a fast Fourier transform (FFT)-like algorithm, to compute the product of the DFT matrix with each sequence of the Legendre pair having any odd length. By utilizing the FFT-like algorithm, we present a low-time complexity algorithm associated with searching for Legendre pairs. Finally, we show numerical results based on the C implementation of the FFT-like algorithm yielding a low time complexity in searching for Legendre pairs as opposed to conventional combinatorial algorithms. This leads us to demonstrate that the proposed FFT-like algorithm significantly accelerates the search for Legendre pairs of orders 45 and 63, achieving at least 99% improvement in speed compared to the conventional algorithms.

Keywords: *Legendre pairs, Matrix analysis, Structured matrices, Sparse and orthogonal factors, Matrix equations, Fast Fourier transforms, Complexity and performance of algorithms*

2020 MSC: 15A23, 15B05, 15B10, 42C10, 65F15, 65F35, 65F40, 65F45, 65T50, 65Y04, 65Y20

☆This work was partially supported by the National Science Foundation award number 2229473.

*Corresponding author

Email addresses: pereras2@erau.edu (Sirani M. Perera), ikotsire@wlu.ca (Ilias S. Kotsireas)

1. Introduction

Hadamard matrices are $n \times n$ matrices H with elements from $\{-1, +1\}$ such that $H \cdot H^t = n \cdot I_n$, where t denotes matrix transposition and I_n is the $n \times n$ identity matrix. These matrices are considered trivial when their order is either $n = 1$ or $n = 2$. Beyond these two trivial orders, a well-known necessary condition for the existence of Hadamard matrices is that $n \equiv 0 \pmod 4$. The sufficiency of this condition is the famous Hadamard conjecture, still open after more than 130 years [1, 2]. Despite several dozens of known constructions for Hadamard matrices, the Hadamard conjecture remains open, mainly because each construction either covers a rather sparse set of orders or fails for several orders. There is a construction that furnishes a promising avenue towards a more structured form of the Hadamard conjecture, i.e., that conjecturally does not fail for any order which is a multiple of four [3]. This construction is based on a combinatorial object called Legendre pairs. Legendre pairs were introduced in [4], where the authors show how to apply the DFT matrix to search for these combinatorial objects. The importance of Legendre pairs is that they can be used to build Hadamard matrices and the conjecture that Legendre pairs of every odd length exist, implies the Hadamard conjecture, albeit via the very specialized structure of Legendre pairs [3]. From the computational point of view, the smallest odd order for which Legendre pairs are not known to exist is 115 [5]. In order to tackle the smallest open case, i.e., $\ell = 115$ with our new algorithm, one would first have to further carefully integrate and incorporate the concept of Djokovic-Kotsireas compression [6] into the current implementation, as well as further optimize the current code. This is planned to be the objective of future work.

In this paper, we explore a new perspective to search for Legendre pairs, i.e., using matrix analysis and linear algebra. We start by exploring the structure of the Legendre pair matrix equation. Structural properties of matrices can be explored to solve many problems in applied sciences and engineering efficiently. These matrices have been studied individually in mathematics, engineering, and computer science disciplines for quite a long time. Within the last two decades, the theories of structured matrices got more attention as they are positioned to bridge diverse disciplines [7, 8, 9, 10, 11, 12]. More specifically, recent work has been done to show that the theories of matrix analysis could be utilized to solve problems in narrowband multi-beam beamforming [13], wideband multi-beam beamforming [14], mutual coupling effects in antenna arrays [15], image processing [16, 17], and reduction and minimization of circuit complexity and power consumption [18, 13, 19, 20, 14, 21] through the derivation of FFT-like algorithms. Challenges remain to develop algorithms for solving problems in diverse disciplines while offering low-complexity, accuracy, and stability at the same time [9, 22]. However, if we could solve linear systems using fast and recursive algorithms, we could obtain numerically stable algorithms with the same order of complexity [23, 24, 25, 26]. Thus, we exploit the structure of the Legendre pair matrix equation to obtain a low-complexity and exact algorithm for searching for these objects.

The FFT is an algorithm that is used to compute the DFT matrix and its inverse efficiently. The FFT algorithm can be used to reduce the brute-force calculation of the $n \times n$ DFT matrix by an $n \times 1$ vector from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log(n))$, where $n = 2^t$ $(t \geq 1)$ [27, 28, 29]. The structure and properties of the DFT matrix yield this complexity reduction. An $n \times n$ circulant is another structured matrix and can be defined using $\mathcal{O}(n)$ as opposed to $\mathcal{O}(n^2)$ parameters. More specifically, circulant matrices are completely determined using the first row (or column) by a cyclic shift of the entries to the right (or down). Furthermore, it is known that the circulant matrices can be diagonalized using the DFT and its conjugate transpose and hence computed by using the FFT algorithms [30, 31, 32, 33]. In this paper, we introduce and incorporate the FFT technique to search for Legendre pairs of ***odd length*** ℓ, efficiently. Thus, we obtain visible speed-ups, especially for odd lengths > 50 while reducing the arithmetic complexity from $\mathcal{O}(\ell^2)$ to $\mathcal{O}(n \log(n))$, where $n > \ell$ is the closest power of 2 to ℓ.

The organization along with the contribution of the paper is as follows. Section 2 presents preliminary results illustrating the use of structured matrices to study the Legendre pair matrix equation and its correlation with the established properties of the two circulant core (2cc) Hadamard matrix. Section 3 presents an algorithm with low arithmetic complexity for computing the product of the DFT matrix with each sequence of the Legendre pair. This algorithm allows for an efficient search for Legendre pairs while minimizing both arithmetic and time complexities. Section 4 presents numerical results demonstrating that the sparse factorization discussed in Section 3 yields an FFT-like algorithm. This, in turn, is utilized in a C implementation, resulting in the algorithm with a low time complexity for searching for Legendre pairs. This algorithm overreaches the time complexity of the existing combinatorial algorithms in searching for Legendre pairs. Finally, Section 5 concludes the paper.

2. Preliminaries and Trivial Properties of the Legendre Pair Matrix Equation via Structured Matrices

This section begins by reviewing the definition of the Legendre pair matrix equation and discussing its basic properties. In the sequel, the Legendre pair matrix equation will be presented as a structured matrix. Legendre pairs are defined via the concept of the periodic autocorrelation function (PAF). Thus, we start the discussion with the definition of the PAF followed by the Legendre pair sequence as stated below.

Definition 2.1. (Periodic autocorrelation function (PAF)) *The periodic autocorrelation function (PAF) of a finite sequence $L_A = [a_0, \ldots, a_{n-1}]$ is defined as:*

$$\mathrm{PAF}_A(s) = \sum_{k=0}^{n-1} a_k a_{k+s}, \ \ s = 0, \ldots, n-1$$

where $k+s$ is taken mod n.

Each $\mathrm{PAF}_A(s)$ value is the inner product of the sequence L_A and its right cyclic shift by s.

Definition 2.2. (Legendre pair) *Let (L_A, L_B) be a pair of sequences of odd length ℓ, consisting of elements from the set $\{-1, +1\}$. (L_A, L_B) is called a Legendre pair if*

$$\mathrm{PAF}(L_A, s) + \mathrm{PAF}(L_B, s) = -2, \ \text{ where } \ s = 1, \ldots, \frac{\ell - 1}{2}.$$

For technical reasons and w.l.o.g. (see [34]) we impose the additional normalization conditions

$$a_1 + \ldots + a_\ell = 1, b_1 + \ldots + b_\ell = 1, \tag{1}$$

where a_i and b_i, for $i = 1, 2, \ldots, \ell$ denote the elements of L_A, L_B respectively.

When (L_A, L_B) forms a Legendre pair of order ℓ and A, B are two $\ell \times \ell$ circulant matrices with their first rows as L_A, L_B respectively, a 2cc Hadamard matrix can be constructed as:

$$H_{2\ell+2} = \left[\begin{array}{cc|c|c} - & - & + \dots + & + \dots + \\ - & + & + \dots + & - \dots - \\ \hline + & + & & \\ \vdots & \vdots & A & B \\ + & + & & \\ \hline + & - & & \\ \vdots & \vdots & B^t & -A^t \\ + & - & & \end{array}\right], \tag{2}$$

where $+$ stands for $+1$ and $-$ stands for -1. Let us recall that the determinant of the Hadamard matrix (2), is given by

$$\det(H_{2\ell+2}) = \pm(2\ell+2)^{\ell+1}. \tag{3}$$

In the sequel, we show a relationship between $\det(H_{2\ell+2})$ and the Legendre pair matrix equation, which will be defined next, to reassure the Property 3 based on the structured matrices. It is important to note that an even-order Legendre pair does not exist. Had there been one, we would have had a Hadamard matrix of order $2\ell+2$, which is congruent to 2 modulo 4. Nevertheless, such a matrix does not exist.

2.1. Fundamentals of the Legendre Pair Matrix Equation

This section starts with the Legendre pair matrix equation, followed by reestablishing the connection between the Legendre pair matrix equation and the 2cc Hadamard matrix via block Gauss-Jordan form and structured matrices.

Lemma 2.3. (Legendre pair matrix equation in [35]) *Let (L_A, L_B) be a Legendre pair of (odd) order ℓ. Let A, B be two $\ell \times \ell$ circulant matrices, whose first rows are L_A, L_B respectively. Then the following matrix equation is satisfied:*

$$AA^t + BB^t = (2\ell+2)I_\ell - 2J_\ell, \tag{4}$$

where I_ℓ denotes the $\ell \times \ell$ identity matrix and J_ℓ denotes the $\ell \times \ell$ all-ones matrix. Moreover, the equation (4) is called the Legendre pair matrix equation.

We denote the right-hand-side matrix of the Legendre pair matrix equation by $M^{(\ell)} = (2\ell + 2)I_\ell - 2J_\ell$, and remark that $M^{(\ell)}$ is a circulant matrix with first row equal to $\underline{r} = [2\ell, \underbrace{-2, \ldots -2}_{\ell-1 \text{ terms}}]$ s.t.

$$M^{(\ell)}_{ij} = \begin{cases} 2\ell & \text{if } i = j, \quad \text{where } i,j = 1,2,\ldots,\ell, \\ -2 & \text{if } i \neq j, \quad \text{where } i,j = 1,2,\ldots,\ell. \end{cases} \tag{5}$$

In the subsequent proposition, we show a relationship linking the determinants of 2cc Hadamard matrices denoted as $H_{2\ell+2}$ in equation (2), with the Legendre pair matrix equation (4), using block Gauss-Jordan form of the 2cc Hadamard matrix.

Proposition 2.4. *Let* $H_{2\ell+2} = \left[\begin{array}{c|c} H_{11} & H_{12} \\ \hline H_{12}^t & H_{22} \end{array}\right]$ *and* A, B *be two* $\ell \times \ell$ *circulant matrices satisfying Equation (4), where* $H_{11} = \begin{bmatrix} -1 & -1 \\ -1 & 1 \end{bmatrix}$, $H_{12} = \begin{bmatrix} 1 \ldots 1 & 1 \ldots 1 \\ 1 \ldots 1 & -1 \ldots -1 \end{bmatrix}_{2,2\ell}$, *and* $H_{22} = \left[\begin{array}{c|c} A & B \\ \hline B^t & -A^t \end{array}\right]$. *Then*

$$\det(H_{2\ell+2}) = 2(\ell+1)^2 \det(M^{(\ell)}). \tag{6}$$

Proof. One can decompose the matrix H_{22} into lower and upper block triangular matrices using the Schur complement of the block A^t of the matrix H_{22} s.t.

$$H_{22} = \left[\begin{array}{c|c} A + B(A^t)^{-1}B^t & B \\ \hline 0 & -A^t \end{array}\right] \left[\begin{array}{c|c} I & 0 \\ \hline -(A^t)^{-1}B^t & I \end{array}\right],$$

where A is the non-singular circulant matrix. Thus, we get

$$\det(H_{22}) = -\det(A + B(A^{-1})^t B^t) \cdot \det(A^t). \tag{7}$$

Similarly, we could decompose $H_{2\ell+2}$ using the nonsingular block circulant matrix H_{22} s.t.

$$H_{2\ell+2} = \left[\begin{array}{c|c} H_{11} - H_{12}H_{22}^{-1}H_{12}^t & H_{12} \\ \hline 0 & H_{22} \end{array}\right] \left[\begin{array}{c|c} I & 0 \\ \hline H_{22}^{-1}H_{12}^t & I \end{array}\right],$$

and obtain $\det(H_{2\ell+2}) = \det(H_{11} - H_{12}H_{22}^{-1}H_{12}^t) \cdot \det(H_{22})$. On the other hand, we can continue the block lower and upper decomposition of H_{22} in order to decompose the matrix H_{22} using a block Gauss-Jordan form of H_{22} s.t.

$$H_{22} = \left[\begin{array}{c|c} I & -B(A^t)^{-1} \\ \hline 0 & I \end{array}\right] \left[\begin{array}{c|c} A + B(A^t)^{-1}B^t & 0 \\ \hline 0 & -A^t \end{array}\right] \left[\begin{array}{c|c} I & 0 \\ \hline -(A^t)^{-1}B^t & I \end{array}\right].$$

Since A is non-singular and H_{22} admits the Gauss-Jordan elimination, we could obtain the inverse of H_{22} s.t.

$$H_{22}^{-1} = \left[\begin{array}{c|c} I & 0 \\ \hline (A^t)^{-1}B^t & I \end{array}\right] \left[\begin{array}{c|c} (A + B(A^t)^{-1}B^t)^{-1} & 0 \\ \hline 0 & -(A^t)^{-1} \end{array}\right] \left[\begin{array}{c|c} I & B(A^t)^{-1} \\ \hline 0 & I \end{array}\right].$$

Thus, from the above, using the circulant matrices A and B generated by L_A and L_B, respectively, and followed by the block matrix multiplication, we get

$$\begin{aligned} &H_{12}H_{22}^{-1}H_{12}^t \\ &= \begin{bmatrix} 2 \dots 2 & 1 \dots 1 \\ 0 \dots 0 & -1 \dots -1 \end{bmatrix}_{2,2\ell} \left[\begin{array}{c|c} (A + B(A^t)^{-1}B^t)^{-1} & 0 \\ \hline 0 & -(A^t)^{-1} \end{array}\right] \begin{bmatrix} 2 \dots 2 & 1 \dots 1 \\ 0 \dots 0 & -1 \dots -1 \end{bmatrix}_{2,2\ell}^t . \end{aligned} \tag{8}$$

Since A and B are circulant matrices derived from the Legendre pair L_A and L_B, respectively, with the normalization condition (1), the sum of each column in the circulant matrices $(A^t)^{-1}B^t$ and $B(A^t)^{-1}$ equals 1. As a result, the entries in the first and last matrices in the factorization (8) will be simplified. Hence, from the above followed by the matrix multiplication, we get

$$H_{12}H_{22}^{-1}H_{12}^t = \begin{bmatrix} \ell & \ell \\ \ell & -\ell \end{bmatrix}.$$

From the above and using the matrix H_{11}, we get

$$\det(H_{11} - H_{12}H_{22}^{-1}H_{12}^t) = -2(\ell+1)^2. \tag{9}$$

Thus, by (7) and (9), we get

$$\det(H_{2\ell+2}) = 2(\ell+1)^2 \det((A + B(A^t)^{-1}B^t)A^t). \tag{10}$$

Since A and B are non-singular circulant matrices, they are commutative, so we have $ABA^{-1} = B$. Thus, we get

$$AA^t + BB^t = AA^t + B(A^{-1})^t B^t A^t. \tag{11}$$

Hence, from (4), (10), and (11), we get the result. □

In the following proposition, we use the structure of the matrix $M^{(\ell)}$ identified via the Legendre pair matrix equation (4) to obtain the determinant of the Legendre pair matrix equation, for any odd length ℓ.

Proposition 2.5. *Let $M^{(\ell)}$ be defined as in (5). Then,*

$$\det(M^{(\ell)}) = 2^{\ell}(\ell+1)^{\ell-1}. \tag{12}$$

Proof. The matrix $M^{(\ell)}$ is circulant. Thus, by using the well-known decomposition of the circulant matrices via the DFT matrices [36, 30, 9, 37, 14], $M^{(\ell)}$ can be expressed via

$$M^{(\ell)} = F_\ell D_\ell F_\ell^*, \tag{13}$$

where $F_\ell = \frac{1}{\sqrt{\ell}}[\omega^{jk}]_{j,k=0}^{\ell-1}$, $\omega = e^{\frac{-2\pi i}{\ell}}$ is a primitive ℓ^{th} root of unity, $i^2=-1$, $D_\ell = \operatorname{diag}(\sqrt{\ell}F_\ell \cdot \underline{r}^t)$, and F_ℓ^* is the conjugate transpose of F_ℓ. Furthermore, the diagonal elements of D_ℓ are eigenvalues of the circulant matrix $M^{(\ell)}$. Let these eigenvalues be represented as λ_j's associated with $M^{(\ell)}$, then these can be explicitly expressed as

$$\lambda_j = 2\ell - 2(\omega^j + \omega^{2j} + \ldots + \omega^{(\ell-1)j}), \tag{14}$$

where $j = 0, 1, \ldots, \ell-1$. Thus, the above equation is simplified into

$$\lambda_j = \begin{cases} 2, & j = 0 \\ 2\ell+2, & j = 1, 2, \ldots, \ell-1. \end{cases} \tag{15}$$

Since F_ℓ and F_ℓ^* are unitary matrices and following equations (13) and (15), we get the result. $\square$

We note that Propositions 2.4 and 2.5 confirm that the determinant of the Hadamard matrix can be calculated using equation (3).

2.2. Eigenvalues and Eigenvectors of the Legendre Pair Matrix Equation

In this section, we continue utilizing the structure of the Legendre pair matrix equation to obtain eigenvalues, eigenvectors, characteristic polynomial, spectral radius, and a bound for the spectrum of the matrices A and B. These properties will be utilized as auxiliary results for the facts on PSD values in Section 3.

Since the Legendre pair matrix equation can be seen as a circulant matrix $M^{(\ell)}$ and the matrix $M^{(\ell)}$ admits the decomposition (13), the normalized eigenvectors of the circulant matrix $M^{(\ell)}$ are the Fourier modes given via $\underline{v}_j = \frac{1}{\sqrt{\ell}}(1, \omega^j, \omega^{2j}, \ldots, \omega^{(\ell-1)j})$, for $j = 0, 1, \ldots, \ell - 1$. Moreover, the eigenvalues of the Legendre pair matrix equation are given via equation (15). Thus, the matrix $M^{(\ell)}$ has the eigenvalues 2 (with multiplicity 1) and $2\ell + 2$ (with multiplicity $\ell - 1$). This leads to the following immediate result based on the characteristic polynomial of the matrix $M^{(\ell)}$.

Corollary 2.6. *Let $M^{(\ell)}$ be defined as in (5). Then its characteristic polynomial, denoted by $P_{M^{(\ell)}}(x)$, is given via*

$$P_{M^{(\ell)}}(x) = (x-2)(x-2\ell-2)^{\ell-1} \tag{16}$$

Proof. This immediately follows from equations (13) and (15). $\square$

Since the eigenvalues of the Legendre pair matrix equation are known, we can state the spectral radius as follows.

Corollary 2.7. *Let $M^{(\ell)}$ be defined as in (5). Then its spectral radius is given via*

$$\rho(M^{(\ell)}) = 2\ell + 2. \tag{17}$$

Proof. This is a direct consequence of the eigenvalues of $M^{(\ell)}$ listed in (15). $\square$

Since the eigenvalues and spectral radius of the Legendre pair matrix equation are stated, we could obtain a bound for the spectrum of eigenvalues based on the Gershgorin disc. Before presenting the next Corollary, let us introduce the Gershgorin disc as follows [38, 39].

Definition 2.8. (Gershgorin disc) *Let A be a complex $n \times n$ matrix with entries a_{ij}. For $i \in \{1, 2, \ldots, n\}$, let R_i be the sum of the absolute values of the non-diagonal entries in the i^{th} row s.t. $R_i = \sum_{j \neq i} |a_{ij}|$. Let $D(a_{ii}, R_i) \subseteq \mathbb{C}$ be a closed disc centered at a_{ii} with radius R_i, then the disc $D(a_{ii}, R_i)$ is called a Gershgorin disc.*

In the following, we use the notation $D(a, R) \subseteq \mathbb{C}$ to denote a closed disc with center $a \in \mathbb{C}$ and radius $R > 0$.

Corollary 2.9. *Let $M^{(\ell)}$ be defined as in (5). Then its eigenvalues lie within the Gershgorin circle $D(2\ell, R) \subseteq \mathbb{C}$, where $R = 2\ell - 2$.*

Proof. This follows immediately from Propositions 2.4 and 2.5, and the Gershgorin circle theorem [38, 39]. $\square$

We utilize the following example to illustrate the above Corollary.

Example 2.10. *Consider the Legendre pair of order $\ell = 45$, given by the sequences L_A and L_B:*

```
L_A:=[-1,-1,1,1,-1,1,1,-1,1,1,-1,-1,1,-1,-1,1,-1,1,1,-1,1,-1,
1,-1,-1,1,-1,-1,1,1,1,1,1,1,-1,-1,1,1,1,1,-1,-1,-1,-1,-1];
```

```
L_B:=[1,-1,-1,1,1,1,-1,-1,1,1,1,1,1,-1,-1,-1,1,1,-1,1,1,1,-1,
-1,1,-1,1,-1,-1,-1,-1,1,-1,-1,-1,1,-1,1,-1,1,-1,1,-1,1,1];
```

In Figure 1, the spectra of the circulant matrices A and B are shown using red points representing the eigenvalues of matrix A and blue points representing the eigenvalues of matrix B, respectively. The light green circle in the figure represents an experimentally determined bound of $11 = \frac{45-1}{4}$ on the radius. Therefore, the theoretically predicted bound of $45 - 1 = 44$ on the radius is not optimal.

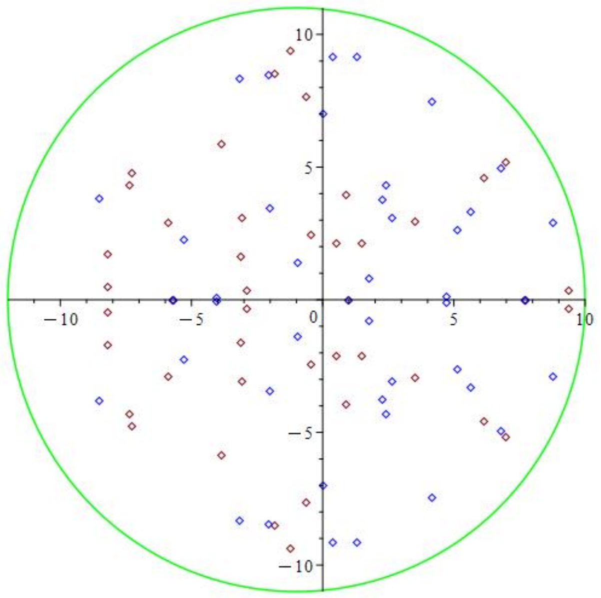

Figure 1: This plot shows the eigenvalues of the Legendre pair of order $\ell = 45$. The red points represent the eigenvalues of matrix A, while blue points indicate the eigenvalues of matrix B. The green circle of radius 11, is a scaled version of the Gershgorin circle centered at $(-1, 0)$.

Continuing from Corollary 2.9, we obtain the following result based on the eigenvalues of the matrices A and B.

Proposition 2.11. *Let (L_A, L_B) be a Legendre pair of (odd) order ℓ. Let A, B be two $\ell \times \ell$ circulant matrices, whose first rows are determined by the sequences L_A, L_B respectively. Then 1 is an eigenvalue of A and also of B.*

Proof. The matrices A and B are $\ell \times \ell$ circulant matrices and hence admit the well-known DFT decomposition s.t.

$$A = F_\ell D_A F_\ell^*, \quad \text{and} \quad B = F_\ell D_B F_\ell^*, \tag{18}$$

where $D_A = \operatorname{diag}(F_\ell \cdot L_A^t)$ and $D_B = \operatorname{diag}(F_\ell \cdot L_B^t)$. Furthermore, the matrices D_A and D_B consist of eigenvalues of the matrices A and B, respectively. Let us denote the eigenvalues of A and B by $(\lambda_A)_j$ and $(\lambda_B)_j$, respectively, for $j = 0, 1, \ldots, \ell - 1$. Thus, using equation (18) allows us to express the entries of D_A and D_B by:

$$(\lambda_A)_j = a_1 + a_2\omega^j + a_3\omega^{2j} + \ldots + a_\ell\omega^{(\ell-1)j}, \text{ and } (\lambda_B)_j = b_1 + b_2\omega^j + b_3\omega^{2j} + \ldots + b_\ell\omega^{(\ell-1)j}.$$

Thus, when $j = 0$, the above equations simplify to

$$(\lambda_A)_0 = a_1 + a_2 + a_3 + \ldots + a_\ell, \text{ and } (\lambda_B)_0 = b_1 + b_2 + b_3 + \ldots + b_\ell. \tag{19}$$

Therefore, based on property (1), i.e., the normalization property yielding the sums in the RHS of the above equation to 1, we get that 1 is an eigenvalue of A and also B. $\square$

3. A Low-complexity Algorithm to Compute PSD Values and Search Legendre Pair

In this section, we first introduce the DFT matrix followed by the FFT and FFTW algorithms. Next, we compute the product of the DFT matrix, with each of the two sequences L_A and L_B of the Legendre pair (L_A, L_B), and we obtain the PSD (power spectral density) as detailed in the Definition 3.1.

Moreover, we state the matrix representation of Bluestein's algorithm in computing the product of the DFT matrix with each of the two sequences L_A and L_B of the Legendre pair (L_A, L_B), as an auxiliary result in searching for Legendre pairs using the FFTW in Section 4. We show that the matrix factorization based on Bluestein's algorithm yields a low arithmetic complexity algorithm while reducing the complexity from $\mathcal{O}(\ell^2)$ to $\mathcal{O}(n\log(n))$, where $n > \ell$ is the closest power of 2 to ℓ.

Finally, we obtain bounds for the PSD values, which can be useful in further pruning the search space of Legendre pairs.

3.1. DFT, FFT, and FFTW

The FFT is the most frequently used algorithm in digital signal processing [40, 41, 42], and is used to efficiently compute the DFT and its inverse. The DFT of a sequence of n inputs $\underline{x} = \{x_k\}_{k=0}^{n-1}$ is a sequence of n outputs $\underline{y} = \{y_k\}_{k=0}^{n-1}$ defined via $\underline{y} = F_n \underline{x}$, where

$$F_n = \frac{1}{\sqrt{n}} [\omega_n^{jk}]_{j,k=0}^{n-1} \tag{20}$$

is the $n \times n$ normalized DFT matrix and $\omega_n = e^{-\frac{2\pi i}{n}}$, which is the superset of ω in (13) as ω defines only the primitive odd, i.e., ℓ^{th}, root of unity. When $n = 2^t$ $(t \geq 1)$, the DFT matrix can be computed using the FFT algorithm [27, 28] because the DFT matrix admits the decomposition

$$F_n = P_n^T \begin{bmatrix} F_{\frac{n}{2}} & \\ & F_{\frac{n}{2}} \end{bmatrix} H_n, \tag{21}$$

where

$$P_n \, \mathbf{x} = \begin{cases} [x_0, x_2, \ldots, x_{n-2}, x_1, x_3, \ldots, x_{n-1}]^T & \text{if } n \text{ is even} \\ [x_0, x_2, \ldots, x_{n-1}, x_1, x_3, \ldots, x_{n-2}]^T & \text{if } n \text{ is odd} \end{cases},$$

$$H_n = \begin{bmatrix} I_{\frac{n}{2}} & I_{\frac{n}{2}} \\ \grave{D}_{\frac{n}{2}} & -\grave{D}_{\frac{n}{2}} \end{bmatrix}$$

is a unitary matrix w.r.t. scaling, $\grave{D}_{\frac{n}{2}} = \text{diag}\left[w_n^l\right]_{l=0}^{\frac{n}{2}-1}$, and I_n is the identity matrix. Hence, the DFT by a vector can be computed using $\mathcal{O}(n \log(n))$ as opposed to $\mathcal{O}(n^2)$ complexity. To reduce the number of multiplications further, one could utilize the normalized DFT matrix s.t. $\tilde{F}_n = \sqrt{n} \;\; F_n$ and its conjugate transpose by F_n^*. As a result, we could further reduce the complexity within the order $\mathcal{O}(n \;\log(n))$ using the decomposition s.t. $\tilde{F}_n = P_n^T \begin{bmatrix} \tilde{F}_{\frac{n}{2}} & \\ & \tilde{F}_{\frac{n}{2}} \end{bmatrix} H_n$.

The difference in complexity of the FFT algorithms from 1965 [27] and the lowest best-known FFT [29] is about 25%, and yet there is a gap between mathematical theories and implementation of the highly optimized FFT packages [43, 44]. Authors in these books implemented FFT in practice by creating the FFTW library, which is a widely used free software library that computes the DFT and its various special cases. Here, we will adopt the FFTW to compute the PSD of the candidate Legendre pairs with the reduction of arithmetic complexity from $\mathcal{O}(\ell^2)$ to $\mathcal{O}(n \;\log(n))$, and also a drastic time complexity reduction in Section 4.

3.2. A FFT-like Algorithm to Compute PSD Values

In this section, we will propose a low arithmetic complexity algorithm for the computation of the set of PSD values, as an alternative to the conventional brute-force approach involving the product of the DFT matrix with each of the two sequences L_A and L_B of the Legendre pair (L_A, L_B). This low arithmetic complexity algorithm will be utilized as a foundation to search for Legendre pairs with a low time complexity algorithm. The algorithm is presented through the matrix representation of Bluestein's algorithm. Before starting the algorithm, let us define the PSD values.

Definition 3.1. (Power spectral density (PSD) values) *Consider two sequences, L_A and L_B, each of odd order ℓ and $m = \frac{\ell-1}{2}$. Let the DFTs (normalized w.r. t. ℓ) of these sequences are denoted by*

$$\text{DFT}(L_A) = [\mu_0, \dots, \mu_{\ell-1}]$$

and

$$\text{DFT}(L_B) = [\nu_0, \dots, \nu_{\ell-1}].$$

Then, the PSD *values of these sequences are defined as:*

$$\text{PSD}(L_A, s) = \Re(\mu_s)^2 + \Im(\mu_s)^2, s = 1, \dots, m$$

and

$$\text{PSD}(L_B, s) = \Re(\nu_s)^2 + \Im(\nu_s)^2, s = 1, \dots, m.$$

The PSD values in the above start from $s = 1$ (omitting $s = 0$) because $\mu_0 = \nu_0 = 1$ is corresponding to the eigenvalues of A and B being 1, and this was shown in Proposition 2.11. In addition, we compute only the first m PSD values because of the symmetry property present in each sequence, i.e., $\text{PSD}(L_A, s) = \text{PSD}(L_A, \ell - s)$ and $\text{PSD}(L_B, s) = \text{PSD}(L_B, \ell - s)$.

Based on the PSD values of the two sequences, we have the PSD invariant property s.t.

$$\text{PSD}(L_A, s) + \text{PSD}(L_B, s) = 2\ell + 2, \ \forall s = 1, \dots, m. \tag{22}$$

In other words, Legendre pairs are characterized by the constancy of PAF (i.e., the PAF values sum to -2) and the constancy of the PSD (i.e., the PSD values sum to $2\ell + 2$). We illustrate these constancy properties of PAF and PSD with the following example:

Example 3.2. *Consider the Legendre pair of order $\ell = 45$ (for which $m = 22$, $2\ell + 2 = 92$)*

```
L_A:=[-1,-1,1,1,-1,1,1,-1,1,1,-1,-1,1,-1,-1,1,-1,1,1,-1,1,-1,
1,-1,-1,1,-1,-1,1,1,1,1,1,1,-1,-1,1,1,1,1,-1,-1,-1,-1,-1];
```

```
L_B:=[1,-1,-1,1,1,1,-1,-1,1,1,1,1,1,-1,-1,-1,1,1,-1,1,1,1,-1,
-1,1,-1,1,-1,-1,-1,-1,1,-1,-1,-1,1,-1,1,-1,1,-1,1,-1,1,1];
```

- *the* PAF *constancy property (with the* PAF *constant -2) is materialized as follows:*

s	1	2	3	4	5	6	7	8	9	10	11
PAF(A, s)	1	-3	13	-11	-7	-3	-3	-3	-3	1	-3
PAF(B, s)	-3	1	-15	9	5	1	1	1	1	-3	1
	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2

s	12	13	14	15	16	17	18	19	20	21	22
PAF(A, s)	-3	-3	1	1	1	1	-3	1	1	1	1
PAF(B, s)	1	1	-3	-3	-3	-3	1	-3	-3	-3	-3
	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2

- *the* PSD *constancy property (with the* PSD *constant $2\ell + 2 = 92$) is materialized as follows:*

```
s, PSD(L_A,s), PSD(L_B,s), PSD(L_A,s) + PSD(L_B,s)
1, 18.78398162, 73.21601830, 92
2, 75.60311755, 16.39688232, 92
3, 75.70559038, 16.29440935, 92
4, 67.60652789, 24.39347193, 92
5, 58.80933390, 33.19066596, 92
6, 58.94004336, 33.05995648, 92
7, 16.29533595, 75.70466380, 92
8, 12.42889922, 79.57110066, 92
9, 32.58359208, 59.41640783, 92
10, 8.393304174, 83.60669564, 92
11, 21.17900710, 70.82099280, 92
```

```
12, 43.12722512, 48.87277485, 92
13, 72.68711906, 19.31288084, 92
14, 89.20751650, 2.792483344, 92
15, 76,          16,          92
16, 88.23332322, 3.766676806, 92
17, 69.87139244, 22.12860750, 92
18, 59.41640784, 32.58359212, 92
19, 49.23886682, 42.76113297, 92
20, 4.797361782, 87.20263814, 92
21, 6.227140824, 85.77285911, 92
22, 6.864911913, 85.13508805, 92
```

We note here that the $\mathrm{PSD}(L_A, s)$ *and* $\mathrm{PSD}(L_B, s)$ *are the square of the magnitude of the eigenvalues of the matrices* A *and* B *generated by the Legendre pair sequences* (L_A, L_B) *in equation (19) for* $s = 1, 2, \ldots, \frac{\ell-1}{2}$.

Remark 3.3. *Numerical experimental evidence gathered for odd values* $\ell < 100$ *suggests that the values of* $\mathrm{PSD}(L_A, 1)$ *and* $\mathrm{PSD}(L_B, 1)$ *are very often a lot larger than the constant* $2\ell + 2$*, therefore this implies that very large swaths of the search space do not contain solutions. As a consequence, these large regions of the search space that do not contain solutions can be pruned efficiently from the search, based on theoretical estimates on the size of the* $\mathrm{PSD}(L_A, 1)$ *and* $\mathrm{PSD}(L_B, 1)$*, as well as (possibly) other* PSD *values.*

Following Example 3.2, one could utilize the FFTW to calculate a set of PSD values instead of brute-force product of the DFT matrix with each of the two sequences L_A and L_B of the Legendre pair (L_A, L_B) while reducing arithmetic and time complexities. The FFTW implementation was done based on the Cooley-Tukey algorithm [27] along with Rader's [45, 46] and Bluestein's [47] FFT algorithms. Rader's and Bluestein's are FFT algorithms computing the DFT of prime and arbitrary sizes by re-expressing the DFT as a cyclic convolution. Since we consider the Legendre pair of odd order ℓ, we state Bluestein's algorithm in the matrix-vector form to compute the product of the DFT matrix with each of the two sequences L_A and L_B of the Legendre pair (L_A, L_B) using $\mathcal{O}(n\ \log(n))$ as opposed to $\mathcal{O}(\ell^2)$ complexity algorithms. This is followed by the definition of the

PSD values concluding that the PSD values can be computed using $\mathcal{O}(n\,\log(n))$ as opposed to the existing $\mathcal{O}(\ell^2)$ complexity algorithms, in the literature.

Proposition 3.4. *Let ℓ be the order of the Legendre pair (L_A, L_B), then the product of the DFT matrix with each of the two sequences L_A and L_B of the Legendre pair (L_A, L_B) can be computed using the following decomposition*

$$\mathrm{DFT}(L_A) = \hat{D}_\ell T_\ell \hat{D}_\ell L_A^t, \tag{23}$$

where $T_\ell = K_{\ell\times n} T_n [K_{\ell\times n}]^t$, $n > \ell$ is the closest power of 2 to ℓ, $\mathrm{DFT} := \sqrt{\ell} F_\ell$,
$K_{\ell\times n} = \left[\begin{array}{c|c} I_\ell & 0_{\ell\times(n-\ell)}\end{array}\right]$, T_n is the symmetric Toeplitz matrix determined via the first column (or row) s.t.
$\underline{t}_{n\times 1} = \left[t_0, t_1, t_2, \ldots, t_{(\ell-1)}, 0, \ldots, 0\right]^t$, $t_j = e^{-\frac{\pi j^2 i}{\ell}}$, *and* $\hat{D}_\ell = \left[\mathrm{diag}(e^{-\frac{\pi k^2 i}{\ell}})\right]_{k=0}^{\ell-1}$.
The $\mathrm{DFT}(L_B)$ is computed using the same factorization as in (23) with the input sequence L_B.

Proof. The symmetric Toeplitz matrix T_ℓ is padded with zeros in order to obtain another symmetric Toeplitz matrix T_n with a length of n, which is the closest power of 2 to ℓ. The first column (or row) of T_n is determined using $\underline{t}_{n\times 1} = \left[t_0, t_1, t_2, \ldots, t_{(\ell-1)}, 0, \ldots, 0\right]^t$. Next, we embed T_n into the circulant matrix R_{2n} s.t. $R_{2n} = \begin{bmatrix} T_n & \tilde{T}_n \\ \tilde{T}_n & T_n \end{bmatrix}$, where a symmetric Toeplitz matrix $\tilde{T}_n$ is determined via its first column (or row) s.t. $\tilde{\underline{t}}_{n\times 1} = \left[t_0, 0, \ldots, 0, t_{(\ell-1)}, t_{(\ell-2)}, \ldots, t_2, t_1\right]^t$. Next, we use 2-FFT (as opposed to the well-known 3-FFTs computing a Toeplitz matrix-vector product) to compute R_{2n} s.t. $R_{2n} = F_{2n}^* D_{2n} F_{2n}$, where $D_{2n} = \mathrm{diag}\left[\tilde{F}_{2n} \underline{t}_{2n\times 1}\right]$, $\underline{t}_{2n\times 1} = \begin{bmatrix} \underline{t} \\ \tilde{\underline{t}} \end{bmatrix}$, and $\tilde{F}_{2n}$ is the normalized DFT matrix. Next, we extract T_n from R_{2n} via $T_n = [J_{2n\times n}]^t R_{2n} J_{2n\times n}$, where $J_{2n\times n} = \begin{bmatrix} I_n \\ \hline 0_n \end{bmatrix}$. Afterwards, we extract T_ℓ from T_n using $K_{\ell\times n}$. Finally, scaling T_ℓ by diagonal matrices $\hat{D}_\ell$ and getting the product of the matrices with each of the two sequences L_A and L_B of the Legendre pair (L_A, L_B) gives the result. □

Remark 3.5.
1. *The above proposition shows that the matrix representation of Bluestein's FFT algorithm is executed via the symmetric Toeplitz matrices T_ℓ [47, 44].*
2. *We recall here that, computing the Toeplitz matrices by a vector using 2-FFTs [14, 37] as opposed to 3-FFTs [9, 15] for an even length s.t. $n = 2^t$ had been proposed.*

3. *One could also pad the odd length ℓ Legendre pair sequences (L_A, L_B) to the closet even length of $n = 2^t$ s.t. $n > \ell$, and execute the product of the DFT matrix with each of the two sequences L_A and L_B of the Legendre pair (L_A, L_B) by using the 2-FFT algorithms in [14, 37]. Since the purpose is also to search for Legendre pairs of odd length ℓ using a low time complexity-based FFTW in C (as in Section 4), which is the Bluestein's FFT algorithm, we proceed with the matrix factorization of Bluestein's FFT in Proposition 23 in order to reduce arithmetic and time complexities.*

Corollary 3.6. *The arithmetic complexity in computing the product of the DFT matrix with each of the two sequences L_A and L_B of the Legendre pair (L_A, L_B) of length ℓ cost $\mathcal{O}(n \log(n))$ operations, where $n > \ell$.*

Proof. Following equation (23), the Toeplitz matrix T_ℓ is computed using T_n followed by 2-FFTs with complexity $\mathcal{O}(n \log(n))$. The cost of computing each diagonal matrix $\hat{D}_\ell$ and D_{2n} by a vector cost $\mathcal{O}(\ell)$ and $\mathcal{O}(n)$, respectively. Thus, the overall cost is $\mathcal{O}(n \log(n))$ as $n > \ell$. □

We recall here that there is also a Winograd FFT algorithm [48] and it minimizes the number of multiplications at the expense of a large number of additions, but this trade-off didn't benefit current processors that need specialized hardware multipliers and was not utilized to implement the FFTW [44].

3.3. Utilize the FFT-like Algorithm for Noise Filtering

The PSD shows the influence of noise signals over a spectrum of frequencies. Thus, the PSD serves as a tool for analyzing noise signals and identifying harmonics, guiding the design of filters. The FFT algorithm is a powerful technique for analyzing the autocorrelation function of a discrete noise signal in order to characterize its PSD. Thus, the FFT-like algorithms are highly efficient in accurately determining the spectral information of dominant noise powers.

In this section, we utilize the proposed FFT-like algorithm with $\ell = 45$ for noise filtering with a simple example. We consider a function of time $f(t) = \cos(\omega_1 t) + \sin(\omega_2 t)$ with frequencies $\omega_1 = 40\pi$ and $\omega_2 = 240\pi$. We then add a large amount of Gaussian white noise to this signal as shown in the first panel of Figure 2. In the second and the third figure panels, we computed the FFT of this noisy signal by using the built-in *fft* command in MATLAB and codes based on Proposition 3.4, respectively. Then, we use the PSD values to filter out noise in both cases. As shown in the bottom two panels, it is possible to clear noise from the signal that has a power below a threshold, which we took as 22 because $m = 22$. Figure 2 displays the resemblance between the numerical results obtained from the proposed algorithm and the *fft* function.

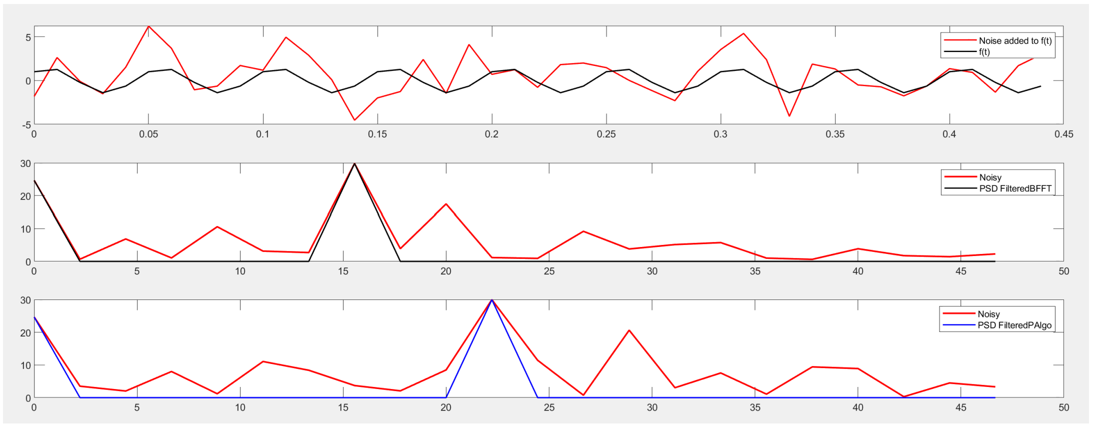

Figure 2: Noise filtering using the built-in *fft* and the proposed algorithm. The top one shows the function $f(t)$ with the added noise. The second and third figure panels show noise signals in the Fourier domain using the built-in FFT (labeled with BFFT) and the proposed algorithm (labeled with PAlgo) followed by the PSD to filter out noise. The proposed algorithm resembles the numerical results with the *fft* function. The x-axis of the top panel represents time and the bottom two panels represent frequency. The y-axis of the bottom two panels shows FFT followed by PSD filtered values based on the proposed algorithm and the *fft* function.

4. Numerical Results for Time Complexity of Searching for Legendre Pairs

In this section, we show that the FFTW-based matrix factorization,i.e., Bluestein's algorithm in Section 3 could be utilized to speed up the process of searching Legendre pairs compared to the existing combinatorial searching algorithms. We illustrate these speedups in the particular cases s.t. $\ell = 45$ and $\ell = 63$.

4.1. Numerical Results for $\ell = 45$

The factorization based on $\ell = 45 = 5 \cdot 9$ allows one to employ 5-decompression (resp. 9-decompression) of a pair of sequences of length 9 (resp. 5), to construct Legendre pairs of order $\ell = 45$. Here is the pair of sequences of length 9, that constitutes the 5-compression of the Legendre pair of order 45 presented in Example 3.2 s.t.

$$[[1, -1, 3, 3, -1, -1, 1, -3, -1], [-1, 1, -1, 3, -1, -1, -3, -1, 5]].$$

Here is a pair of sequences of length 5, that constitutes the 9-compression of the Legendre pair of order 45 presented in Example 3.2 s.t.

$$[[1,-3,3,3,-3],[1,3,-3,-3,3]].$$

Our currently available C code implementations find Legendre pairs of order $\ell = 45$ in about 20 minutes, while the FFTW-enabled C codes find Legendre pairs of order $\ell = 45$ in less than 10 seconds. Thus, the proposed FFT-like algorithm enables the search for Legendre pairs of order $\ell = 45$ with a 99% efficiency, outperforming existing methods.

4.2. Numerical Results for $\ell = 63$

The factorization $\ell = 63 = 7 \cdot 9$ allows one to employ 7-decompression [6] of a pair of sequences of length 9, to construct Legendre pairs of order $\ell = 63$. In particular, we employ two copies of the same sequence of length 9 s.t.

$$S_9 = [-7,1,1,1,1,1,1,1,1].$$

This sequence has the following properties:

$$\text{PAF}(S_9,s) = -7, s = 1,\ldots,4 \text{ and } \text{PSD}(S_9,s) = 64, s = 1,\ldots,4,$$

i.e., it satisfies the hypotheses of Proposition 1 in [34]. Since the first element of S_9 has a unique 7-decompressed, the complexity of the 7-decompression code required in order to construct Legendre pairs of order $\ell = 63$ is dominated by the remaining 8 elements, each one of which can be 7-decompressed in $\binom{7}{4} = 35$ ways. Therefore, we need to compute $35^8 \approx 2^{41}$ DFTs of $\{-1,+1\}$-sequences of length 63. For each one of these 2^{41} DFTs, we perform a lookup in the computed DFT vector of length 63, in order to see whether all the $\frac{63-1}{2} = 31$ PSD values are less than the PSD constant $2 \cdot 63 + 2 = 128$.

Using the FFTW library in C:

1. 4 Legendre pairs are found in less than 60 minutes, with about 0.2% of the entire space traversed;
2. 56 Legendre pairs are found in 12 hours, with about 3% of the entire space traversed.

From the above data, one can conclude that the entire search space can be traversed super fast with our novel implementation, while the existing C implementation for Legendre pairs of order $\ell = 63$ did not even produce any results i.e. not even one Legendre pair after 24 hours.

5. Conclusion

In this paper, we provided novel perspectives on Legendre pairs, using concepts from matrix analysis and linear algebra. These perspectives can be used (with great benefits) in current computational schemes to search for Legendre pairs. The structured matrices perspectives on Legendre pairs make use of the Legendre pair matrix equation, to investigate its properties. We introduced a structured matrix approach to obtain a low arithmetic complexity algorithm for computing the product of the DFT matrix with each sequence of the Legendre pair, regardless of their odd lengths. This method enables efficient computation of power spectral density values, and hence, search for Legendre pairs with reduced time complexity. Finally, we showed numerical results based on the C implementation of FFTW in searching for Legendre pairs while attaining a low time complexity algorithm so that the proposed technique excels the conventional combinatorial algorithms in searching Legendre pairs. More specifically, we had shown that the proposed FFT-like algorithm significantly accelerates the search for Legendre pairs of orders 45 and 63, achieving at least 99% improvement in speed compared to conventional algorithms, in the literature.

Acknowledgement

We appreciate the valuable comments and feedback provided by the editor and the reviewers to enhance the composition of the manuscript.

Appendix: Legendre Pairs database for some small lengths

In here, we put a list of Legendre pairs for readers to test the corresponding problems for small lengths.

```
ell = 3:
A:=CirculantMatrix([1,1,-1]);
B:=CirculantMatrix([1,1,-1]);

ell = 5:
A:=CirculantMatrix([-1, -1, 1, 1, 1]);
B:=CirculantMatrix([-1, 1, -1, 1, 1]);

ell = 7:
A:=CirculantMatrix([-1, -1, 1, -1, 1, 1, 1]);
B:=CirculantMatrix([-1, -1, 1, -1, 1, 1, 1]);

ell = 9:
A:=CirculantMatrix([-1, -1, -1, 1, -1, 1, 1, 1, 1]);
B:=CirculantMatrix([-1, -1, 1, 1, -1, 1, -1, 1, 1]);

ell = 11:
A:=CirculantMatrix([-1, -1, -1, -1, 1, 1, -1, 1, 1, 1, 1]);
B:=CirculantMatrix([-1, -1, 1, 1, -1, 1, -1, 1, -1, 1, 1]);

ell = 13:
A:=CirculantMatrix([-1, -1, -1, -1, 1, -1, 1, 1, 1, -1, 1, 1, 1]);
B:=CirculantMatrix([-1, -1, 1, -1, 1, 1, -1, -1, 1, 1, 1, -1, 1]);

ell = 15:
A:=CirculantMatrix([-1, -1, -1, -1, -1, 1, 1, -1, 1, 1, -1, 1, 1, 1, 1]);
```

```
B:=CirculantMatrix([-1, -1, 1, -1, 1, -1, 1, -1, 1, 1, -1, -1, 1, 1, 1]);


ell = 17:
A:=CirculantMatrix([-1, -1, -1, -1, -1, 1, -1, 1, 1, 1, -1, 1, 1, -1, 1, 1, 1]);
B:=CirculantMatrix([-1, -1, -1, 1, -1, 1, 1, -1, 1, -1, 1, 1, -1, -1, 1, 1, 1]);


ell = 19:
A:=CirculantMatrix([-1, -1, -1, -1, -1, -1, 1, 1, 1, -1, 1, 1, -1, 1, 1, -1, 1, 1, 1]);
B:=CirculantMatrix([-1, -1, -1, 1, 1, -1, -1, 1, -1, 1, 1, -1, 1, -1, 1, -1, 1, 1, 1]);
```

Appendix: perform 7-decompression for $S_9 = [-7, 1, 1, 1, 1, 1, 1, 1, 1]$, to find for $LP(63)$.

```
#include <stdio.h>
#include <stdlib.h>
#include <fftw3.h>
#include <math.h>
int main(int argc, char *argv[])
{
int ii, N=63;
fftw_complex *in, *out;
fftw_plan p;

in = (fftw_complex*) fftw_malloc(sizeof(fftw_complex) * N);
out = (fftw_complex*) fftw_malloc(sizeof(fftw_complex) * N);
p = fftw_plan_dft_1d(N, in, out, FFTW_FORWARD, FFTW_ESTIMATE);

int i, v = 63;

int Op1s[] = {0,2,3,4,5,6,7,8,9};

int Om1s[] = {0};

int Op3s[] = {0};

int Om3s[] = {0};

int Op5s[] = {0};

int Om5s[] = {0};

int xm1[1], xp1[9], xm3[1], xp3[1], xm5[1], xp5[1], A[64];

float  PSD1, PSD2, PSD3, PSD4, PSD5, PSD6, PSD7, PSD8, PSD9, PSD10,
PSD11, PSD12, PSD13, PSD14, PSD15, PSD16, PSD17, PSD18, PSD19, PSD20,
PSD21, PSD22, PSD23, PSD24, PSD25, PSD26, PSD27, PSD28, PSD29, PSD30, PSD31;

A[1] = -1; A[10] = -1; A[19] = -1; A[28] = -1; A[37] = -1; A[46] = -1; A[55] = -1;

for (xp1[1] = 1; xp1[1] <= 35; xp1[1]++)
for (xp1[2] = 1; xp1[2] <= 35; xp1[2]++)
for (xp1[3] = 1; xp1[3] <= 35; xp1[3]++)
for (xp1[4] = 1; xp1[4] <= 35; xp1[4]++)
for (xp1[5] = 1; xp1[5] <= 35; xp1[5]++)
for (xp1[6] = 1; xp1[6] <= 35; xp1[6]++)
for (xp1[7] = 1; xp1[7] <= 35; xp1[7]++)
for (xp1[8] = 1; xp1[8] <= 35; xp1[8]++)
        {
for(i=1; i<=8; i++)
   switch (xp1[i]) {
case 1  : A[Op1s[i]] = -1 ; A[Op1s[i]+9] = -1 ; A[Op1s[i]+18] = -1 ; A[Op1s[i]+27] = 1 ; A[Op1s[i]+36] = 1 ; A[Op1s[i]+45] = 1 ; A[Op1s[i]+54] = 1 ; break;
case 2  : A[Op1s[i]] = -1 ; A[Op1s[i]+9] = -1 ; A[Op1s[i]+18] = 1 ; A[Op1s[i]+27] = -1 ; A[Op1s[i]+36] = 1 ; A[Op1s[i]+45] = 1 ; A[Op1s[i]+54] = 1 ; break;
case 3  : A[Op1s[i]] = -1 ; A[Op1s[i]+9] = -1 ; A[Op1s[i]+18] = 1 ; A[Op1s[i]+27] = 1 ; A[Op1s[i]+36] = -1 ; A[Op1s[i]+45] = 1 ; A[Op1s[i]+54] = 1 ; break;
case 4  : A[Op1s[i]] = -1 ; A[Op1s[i]+9] = -1 ; A[Op1s[i]+18] = 1 ; A[Op1s[i]+27] = 1 ; A[Op1s[i]+36] = 1 ; A[Op1s[i]+45] = -1 ; A[Op1s[i]+54] = 1 ; break;
case 5  : A[Op1s[i]] = -1 ; A[Op1s[i]+9] = -1 ; A[Op1s[i]+18] = 1 ; A[Op1s[i]+27] = 1 ; A[Op1s[i]+36] = 1 ; A[Op1s[i]+45] = 1 ; A[Op1s[i]+54] = -1 ; break;
case 6  : A[Op1s[i]] = -1 ; A[Op1s[i]+9] = 1 ; A[Op1s[i]+18] = -1 ; A[Op1s[i]+27] = -1 ; A[Op1s[i]+36] = 1 ; A[Op1s[i]+45] = 1 ; A[Op1s[i]+54] = 1 ; break;
case 7  : A[Op1s[i]] = -1 ; A[Op1s[i]+9] = 1 ; A[Op1s[i]+18] = -1 ; A[Op1s[i]+27] = 1 ; A[Op1s[i]+36] = -1 ; A[Op1s[i]+45] = 1 ; A[Op1s[i]+54] = 1 ; break;
case 8  : A[Op1s[i]] = -1 ; A[Op1s[i]+9] = 1 ; A[Op1s[i]+18] = -1 ; A[Op1s[i]+27] = 1 ; A[Op1s[i]+36] = 1 ; A[Op1s[i]+45] = -1 ; A[Op1s[i]+54] = 1 ; break;
case 9  : A[Op1s[i]] = -1 ; A[Op1s[i]+9] = 1 ; A[Op1s[i]+18] = -1 ; A[Op1s[i]+27] = 1 ; A[Op1s[i]+36] = 1 ; A[Op1s[i]+45] = 1 ; A[Op1s[i]+54] = -1 ; break;
case 10  : A[Op1s[i]] = -1 ; A[Op1s[i]+9] = 1 ; A[Op1s[i]+18] = 1 ; A[Op1s[i]+27] = -1 ; A[Op1s[i]+36] = -1 ; A[Op1s[i]+45] = 1 ; A[Op1s[i]+54] = 1 ; break;
case 11  : A[Op1s[i]] = -1 ; A[Op1s[i]+9] = 1 ; A[Op1s[i]+18] = 1 ; A[Op1s[i]+27] = -1 ; A[Op1s[i]+36] = 1 ; A[Op1s[i]+45] = -1 ; A[Op1s[i]+54] = 1 ; break;
case 12  : A[Op1s[i]] = -1 ; A[Op1s[i]+9] = 1 ; A[Op1s[i]+18] = 1 ; A[Op1s[i]+27] = -1 ; A[Op1s[i]+36] = 1 ; A[Op1s[i]+45] = 1 ; A[Op1s[i]+54] = -1 ; break;
case 13  : A[Op1s[i]] = -1 ; A[Op1s[i]+9] = 1 ; A[Op1s[i]+18] = 1 ; A[Op1s[i]+27] = 1 ; A[Op1s[i]+36] = -1 ; A[Op1s[i]+45] = -1 ; A[Op1s[i]+54] = 1 ; break;
case 14  : A[Op1s[i]] = -1 ; A[Op1s[i]+9] = 1 ; A[Op1s[i]+18] = 1 ; A[Op1s[i]+27] = 1 ; A[Op1s[i]+36] = -1 ; A[Op1s[i]+45] = 1 ; A[Op1s[i]+54] = -1 ; break;
case 15  : A[Op1s[i]] = -1 ; A[Op1s[i]+9] = 1 ; A[Op1s[i]+18] = 1 ; A[Op1s[i]+27] = 1 ; A[Op1s[i]+36] = 1 ; A[Op1s[i]+45] = -1 ; A[Op1s[i]+54] = -1 ; break;
case 16  : A[Op1s[i]] = 1 ; A[Op1s[i]+9] = -1 ; A[Op1s[i]+18] = -1 ; A[Op1s[i]+27] = -1 ; A[Op1s[i]+36] = 1 ; A[Op1s[i]+45] = 1 ; A[Op1s[i]+54] = 1 ; break;
case 17  : A[Op1s[i]] = 1 ; A[Op1s[i]+9] = -1 ; A[Op1s[i]+18] = -1 ; A[Op1s[i]+27] = 1 ; A[Op1s[i]+36] = -1 ; A[Op1s[i]+45] = 1 ; A[Op1s[i]+54] = 1 ; break;
case 18  : A[Op1s[i]] = 1 ; A[Op1s[i]+9] = -1 ; A[Op1s[i]+18] = -1 ; A[Op1s[i]+27] = 1 ; A[Op1s[i]+36] = 1 ; A[Op1s[i]+45] = -1 ; A[Op1s[i]+54] = 1 ; break;
case 19  : A[Op1s[i]] = 1 ; A[Op1s[i]+9] = -1 ; A[Op1s[i]+18] = -1 ; A[Op1s[i]+27] = 1 ; A[Op1s[i]+36] = 1 ; A[Op1s[i]+45] = 1 ; A[Op1s[i]+54] = -1 ; break;
case 20  : A[Op1s[i]] = 1 ; A[Op1s[i]+9] = -1 ; A[Op1s[i]+18] = 1 ; A[Op1s[i]+27] = -1 ; A[Op1s[i]+36] = -1 ; A[Op1s[i]+45] = 1 ; A[Op1s[i]+54] = 1 ; break;
case 21  : A[Op1s[i]] = 1 ; A[Op1s[i]+9] = -1 ; A[Op1s[i]+18] = 1 ; A[Op1s[i]+27] = -1 ; A[Op1s[i]+36] = 1 ; A[Op1s[i]+45] = -1 ; A[Op1s[i]+54] = 1 ; break;
case 22  : A[Op1s[i]] = 1 ; A[Op1s[i]+9] = -1 ; A[Op1s[i]+18] = 1 ; A[Op1s[i]+27] = -1 ; A[Op1s[i]+36] = 1 ; A[Op1s[i]+45] = 1 ; A[Op1s[i]+54] = -1 ; break;
case 23  : A[Op1s[i]] = 1 ; A[Op1s[i]+9] = -1 ; A[Op1s[i]+18] = 1 ; A[Op1s[i]+27] = 1 ; A[Op1s[i]+36] = -1 ; A[Op1s[i]+45] = -1 ; A[Op1s[i]+54] = 1 ; break;
case 24  : A[Op1s[i]] = 1 ; A[Op1s[i]+9] = -1 ; A[Op1s[i]+18] = 1 ; A[Op1s[i]+27] = 1 ; A[Op1s[i]+36] = -1 ; A[Op1s[i]+45] = 1 ; A[Op1s[i]+54] = -1 ; break;
case 25  : A[Op1s[i]] = 1 ; A[Op1s[i]+9] = -1 ; A[Op1s[i]+18] = 1 ; A[Op1s[i]+27] = 1 ; A[Op1s[i]+36] = 1 ; A[Op1s[i]+45] = -1 ; A[Op1s[i]+54] = -1 ; break;
case 26  : A[Op1s[i]] = 1 ; A[Op1s[i]+9] = 1 ; A[Op1s[i]+18] = -1 ; A[Op1s[i]+27] = -1 ; A[Op1s[i]+36] = -1 ; A[Op1s[i]+45] = 1 ; A[Op1s[i]+54] = 1 ; break;
case 27  : A[Op1s[i]] = 1 ; A[Op1s[i]+9] = 1 ; A[Op1s[i]+18] = -1 ; A[Op1s[i]+27] = -1 ; A[Op1s[i]+36] = 1 ; A[Op1s[i]+45] = -1 ; A[Op1s[i]+54] = 1 ; break;
case 28  : A[Op1s[i]] = 1 ; A[Op1s[i]+9] = 1 ; A[Op1s[i]+18] = -1 ; A[Op1s[i]+27] = -1 ; A[Op1s[i]+36] = 1 ; A[Op1s[i]+45] = 1 ; A[Op1s[i]+54] = -1 ; break;
case 29  : A[Op1s[i]] = 1 ; A[Op1s[i]+9] = 1 ; A[Op1s[i]+18] = -1 ; A[Op1s[i]+27] = 1 ; A[Op1s[i]+36] = -1 ; A[Op1s[i]+45] = -1 ; A[Op1s[i]+54] = 1 ; break;
case 30  : A[Op1s[i]] = 1 ; A[Op1s[i]+9] = 1 ; A[Op1s[i]+18] = -1 ; A[Op1s[i]+27] = 1 ; A[Op1s[i]+36] = -1 ; A[Op1s[i]+45] = 1 ; A[Op1s[i]+54] = -1 ; break;
case 31  : A[Op1s[i]] = 1 ; A[Op1s[i]+9] = 1 ; A[Op1s[i]+18] = -1 ; A[Op1s[i]+27] = 1 ; A[Op1s[i]+36] = 1 ; A[Op1s[i]+45] = -1 ; A[Op1s[i]+54] = -1 ; break;
case 32  : A[Op1s[i]] = 1 ; A[Op1s[i]+9] = 1 ; A[Op1s[i]+18] = 1 ; A[Op1s[i]+27] = -1 ; A[Op1s[i]+36] = -1 ; A[Op1s[i]+45] = -1 ; A[Op1s[i]+54] = 1 ; break;
case 33  : A[Op1s[i]] = 1 ; A[Op1s[i]+9] = 1 ; A[Op1s[i]+18] = 1 ; A[Op1s[i]+27] = -1 ; A[Op1s[i]+36] = -1 ; A[Op1s[i]+45] = 1 ; A[Op1s[i]+54] = -1 ; break;
case 34  : A[Op1s[i]] = 1 ; A[Op1s[i]+9] = 1 ; A[Op1s[i]+18] = 1 ; A[Op1s[i]+27] = -1 ; A[Op1s[i]+36] = 1 ; A[Op1s[i]+45] = -1 ; A[Op1s[i]+54] = -1 ; break;
case 35  : A[Op1s[i]] = 1 ; A[Op1s[i]+9] = 1 ; A[Op1s[i]+18] = 1 ; A[Op1s[i]+27] = 1 ; A[Op1s[i]+36] = -1 ; A[Op1s[i]+45] = -1 ; A[Op1s[i]+54] = -1 ; break;
  }

for (ii = 0; ii < N; ++ii){
  in[ii][0]=A[ii+1];
  in[ii][1]=0;
}
fftw_execute(p);

if ( out[1][0]*out[1][0] + out[1][1]*out[1][1] > 128 ) continue;
if ( out[2][0]*out[2][0] + out[2][1]*out[2][1] > 128 ) continue;
if ( out[3][0]*out[3][0] + out[3][1]*out[3][1] > 128 ) continue;
if ( out[4][0]*out[4][0] + out[4][1]*out[4][1] > 128 ) continue;
if ( out[5][0]*out[5][0] + out[5][1]*out[5][1] > 128 ) continue;
if ( out[6][0]*out[6][0] + out[6][1]*out[6][1] > 128 ) continue;
if ( out[7][0]*out[7][0] + out[7][1]*out[7][1] > 128 ) continue;
if ( out[8][0]*out[8][0] + out[8][1]*out[8][1] > 128 ) continue;
if ( out[9][0]*out[9][0] + out[9][1]*out[9][1] > 128 ) continue;
if ( out[10][0]*out[10][0] + out[10][1]*out[10][1] > 128 ) continue;
if ( out[11][0]*out[11][0] + out[11][1]*out[11][1] > 128 ) continue;
if ( out[12][0]*out[12][0] + out[12][1]*out[12][1] > 128 ) continue;
if ( out[13][0]*out[13][0] + out[13][1]*out[13][1] > 128 ) continue;
if ( out[14][0]*out[14][0] + out[14][1]*out[14][1] > 128 ) continue;
if ( out[15][0]*out[15][0] + out[15][1]*out[15][1] > 128 ) continue;
if ( out[16][0]*out[16][0] + out[16][1]*out[16][1] > 128 ) continue;
if ( out[17][0]*out[17][0] + out[17][1]*out[17][1] > 128 ) continue;
if ( out[18][0]*out[18][0] + out[18][1]*out[18][1] > 128 ) continue;
if ( out[19][0]*out[19][0] + out[19][1]*out[19][1] > 128 ) continue;
if ( out[20][0]*out[20][0] + out[20][1]*out[20][1] > 128 ) continue;
if ( out[21][0]*out[21][0] + out[21][1]*out[21][1] > 128 ) continue;
if ( out[22][0]*out[22][0] + out[22][1]*out[22][1] > 128 ) continue;
if ( out[23][0]*out[23][0] + out[23][1]*out[23][1] > 128 ) continue;
if ( out[24][0]*out[24][0] + out[24][1]*out[24][1] > 128 ) continue;
if ( out[25][0]*out[25][0] + out[25][1]*out[25][1] > 128 ) continue;
if ( out[26][0]*out[26][0] + out[26][1]*out[26][1] > 128 ) continue;
if ( out[27][0]*out[27][0] + out[27][1]*out[27][1] > 128 ) continue;
if ( out[28][0]*out[28][0] + out[28][1]*out[28][1] > 128 ) continue;
if ( out[29][0]*out[29][0] + out[29][1]*out[29][1] > 128 ) continue;
if ( out[30][0]*out[30][0] + out[30][1]*out[30][1] > 128 ) continue;
if ( out[31][0]*out[31][0] + out[31][1]*out[31][1] > 128 ) continue;

printf("%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d%d\n",
(int)rint(out[1][0]*out[1][0] + out[1][1]*out[1][1]) % 10, (int)rint(out[2][0]*out[2][0] +
out[2][1]*out[2][1]) % 10, (int)rint(out[3][0]*out[3][0] + out[3][1]*out[3][1]) % 10,
(int)rint(out[4][0]*out[4][0] + out[4][1]*out[4][1]) % 10, (int)rint(out[5][0]*out[5][0] +
out[5][1]*out[5][1]) % 10, (int)rint(out[6][0]*out[6][0] + out[6][1]*out[6][1]) % 10,
(int)rint(out[8][0]*out[8][0] + out[8][1]*out[8][1]) % 10, (int)rint(out[9][0]*out[9][0] +
out[9][1]*out[9][1]) % 10, (int)rint(out[10][0]*out[10][0] + out[10][1]*out[10][1]) % 10, (int)rint(out[11][0]*out[11][0] + out[11][1]*out[11][1]) % 10, (int)rint(out[12][0]*out[12][0] +
out[12][1]*out[12][1]) % 10, (int)rint(out[13][0]*out[13][0] + out[13][1]*out[13][1]) % 10, (int)rint(out[15][0]*out[15][0] + out[15][1]*out[15][1]) % 10, (int)rint(out[16][0]*out[16][0] +
out[16][1]*out[16][1]) % 10, (int)rint(out[17][0]*out[17][0] + out[17][1]*out[17][1]) % 10, (int)rint(out[18][0]*out[18][0] + out[18][1]*out[18][1]) % 10, (int)rint(out[19][0]*out[19][0] +
out[19][1]*out[19][1]) % 10, (int)rint(out[20][0]*out[20][0] + out[20][1]*out[20][1]) % 10, (int)rint(out[22][0]*out[22][0] + out[22][1]*out[22][1]) % 10, (int)rint(out[23][0]*out[23][0] +
out[23][1]*out[23][1]) % 10, (int)rint(out[24][0]*out[24][0] + out[24][1]*out[24][1]) % 10, (int)rint(out[25][0]*out[25][0] + out[25][1]*out[25][1]) % 10, (int)rint(out[26][0]*out[26][0] +
out[26][1]*out[26][1]) % 10, (int)rint(out[27][0]*out[27][0] + out[27][1]*out[27][1]) % 10, (int)rint(out[29][0]*out[29][0] + out[29][1]*out[29][1]) % 10, (int)rint(out[30][0]*out[30][0] +
out[30][1]*out[30][1]) % 10, (int)rint(out[31][0]*out[31][0] + out[31][1]*out[31][1]) % 10);

}
  fftw_destroy_plan(p);  fftw_free(in);  fftw_free(out);  return(0);
}
```

References

[1] M. Hall, Jr., Combinatorial theory, 2nd Edition, Wiley Classics Library, John Wiley & Sons, Inc., New York, 1998, a Wiley-Interscience Publication.

[2] D. R. Stinson, Combinatorial designs - constructions and analysis, Springer, 2004.

[3] I. S. Kotsireas, Structured Hadamard Conjecture, in: J. M. Borwein, I. E. Shparlinski, W. Zudilin (Eds.), Number Theory and Related Fields, In Memory of Alf van der Poorten, no. Theory, Springer, 2013, pp. 215–227. `doi:10.1007/978-1-4614-6642-0\_11`.
URL `https://doi.org/10.1007/978-1-4614-6642-0_11`

[4] R. J. Fletcher, M. Gysin, J. Seberry, Application of the discrete Fourier transform to the search for generalised Legendre pairs and Hadamard matrices, Australas. J. Combin. 23 (2001) 75–86.

[5] I. S. Kotsireas, C. Koutschan, D. A. Bulutoglu, D. M. Arquette, J. S. Turner, K. J. Ryan, Legendre pairs of lengths 0 (mod 5), Special Matrices 11 (2023).
URL `https://api.semanticscholar.org/CorpusID:265142743`

[6] D. Z. Dokovic, I. S. Kotsireas, Compression of periodic complementary sequences and applications, Des. Codes Cryptogr. 74 (2) (2015) 365–377. `doi:10.1007/s10623-013-9862-z`.
URL `https://doi.org/10.1007/s10623-013-9862-z`

[7] V. Olshevsky, Structured Matrices in Mathematics, Computer Science, and Engineering, Contemporary Mathematics Series, 280 and 281, American Mathematical Society, USA, 2001.

[8] V. Olshevsky, Fast Algorithms for Structured Matrices: Theory and Applications, Contemporary Mathematics,323, American Mathematical Society, USA, 2003.

[9] T. Kailath, A. Sayed, Fast Reliable Algorithms for Matrices with Structure, SIAM Publications, Philadelphia, USA, 1999.

[10] M. Benzi, V. Simoncini(eds), Exploiting Hidden Structure in Matrix Computations: Algorithms and Applications, Springer, Cham, 2016.

[11] D. A. Bini, F. D. Benedetto, E. Tyrtyshnikov, M. V. Barel, Structured Matrices in Numerical Linear Algebra: Analysis, Algorithms, and Applications, Springer INdAM Series, Springer Publications, Switzerland, 2019.

[12] A. Paulraj, V. Roychowdhury, C. D. Schaper, Communications, computation, control and signal processing, in: a tribute to Thomas Kailath, 1997.

[13] S. M. Perera, A. Madanayake, R. Cintra, Radix-2 self-recursive algorithms for vandermonde-type matrices and true-time-delay multi-beam antenna arrays, IEEE Access 8 (2020) 25498–25508.

[14] S. M. Perera, L. Lingsch, A. Madanayake, S. Mandal, N. Mastronardi, Fast dvm algorithm for wideband time-delay multi-beam beamformers, the IEEE Transactions on Signal Processing 70 (5913-5925) (2022).

[15] S. M. Perera, L. Lingsch, A. Madanayake, L. Belostotski, A low-complexity algorithm to digitally uncouple the mutual coupling effect in antenna arrays, in: submitted to the Journal of Computational and Applied Mathematics, 2023.

[16] S. M. Perera, L. Lingsch, Sparse matrix-based low-complexity, recursive, radix-2 algorithms for discrete sine transforms, IEEE Access 9 (2021) 141181–141198.

[17] S. M. Perera, J. Liu, Lowest complexity self recursive radix-2 dct ii/iii algorithms, SIAM J. Matrix Analysis and Applications 39 (2) (2018) 664–682.

[18] S. M. Perera, D. Silverio, A. Ogle, Efficient split-radix and radix-4 dct algorithms and applications, in: in Proc. the Analysis of Experimental Algorithms, Lecture Notes in Computer Science 11544, 2019, pp. 184–201.

[19] S. M. Perera, J. Liu, Complexity reduction, self/completely recursive, radix-2 dct i/iv algorithms, Journal of Computational Applied Mathematics 379 (112936) (2020) 1–16.

[20] D. F. G. Coelho, R. Cintra, A. Madanayake, S. M. Perera, Low-complexity scaling methods for dct-ii approximation, IEEE Transactions on Signal Processing 69 (2021) 4557–4566.

[21] S. M. Perera, Signal flow graph approach to efficient and forward stable dst algorithms, Linear Algebra and Its Applications 542 (2018) 360–390.

[22] N. J. Higham, Accuracy and Stability of Numerical Algorithms, SIAM, Philadelphia, USA, 1996.

[23] J. Demmel, I. Dumitriu, O. Holtz, Fast linear algebra is stable, Numerische Mathematik 108 (2007) 59–91.

[24] A. Borodin, T. Munro, The Computational Complexity of Algebraic and Numeric Problems, American Elsevier, Amsterdam, 1975.

[25] P. Bürgisser, M. Clausen, M. A. Shokrollahi, Algebraic Complexity Theory, Springer, Berlin, 1997.

[26] D. Heller, A survey of parallel algorithms in numerical linear algebra, SIAM Rev. 20 (1978) 740–777.

[27] J. W. Cooley, J. W. Tukey, An algorithm for the machine calculation of complex fourier series, Math. Comp. 19 (1965) 297–301.

[28] R. Yavne, An economical method for calculating the discrete fourier transform, in: in Proc. AFIPS Fall Joint Computer Conf. 33, 1968, pp. 115–125.

[29] S. G. Johnson, F. M., A modified split-radix fft with fewer arithmetic operations, IEEE Trans. 55 (1) (2007) 111–119.

[30] G. Golub, C. V. Loan, Matrix Computations 4th ed., The Johns Hopkins University Press, Baltimore, MD, 2013.

[31] P. J. David, Circulant Matrices (Pure and applied mathematics) 1st Ed, Wiley, New York, 1979.

[32] I. Gohberg, V. Olshevsky, Complexity of multiplication with vectors for structured matrices, Linear Algebra Appl. 202 (1994) 163–192.

[33] I. Gohberg, V. Olshevsky, Fast algorithms with preprocessing for matrix-vector multiplication problems. journal of complexity, Linear Algebra Appl. 10 (4) (1994) 411–427.

[34] I. Kotsireas, C. Koutschan, Legendre pairs of lengths $\ell \equiv 0 \pmod 3$, J. Combin. Des. 29 (12) (2021) 870–887. `doi:10.1002/jcd.21806`.
URL `https://doi.org/10.1002/jcd.21806`

[35] M. Chiarandini, I. S. Kotsireas, C. Koukouvinos, L. Paquete, Heuristic algorithms for Hadamard matrices with two circulant cores, Theoret. Comput. Sci. 407 (1-3) (2008) 274–277.

[36] P. J. Davis, Circulant Matrices, Wiley, New York, 1970.

[37] D. A. Bini, Matrix structures in queuing models, in: In: Benzi M., Simoncini V. (eds), Exploiting Hidden Structure in Matrix Computations: Algorithms and Applications, Lecture Notes in Mathematics, 2173, 2016, pp. 65–160.

[38] S. Gerschgorin, über die abgrenzung der eigenwerte einer matrix, Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na 6 (1931) 749–754.

[39] R. S. Varga, Gersgorin and His Circles, Springer-Verlag, Berlin, 2004.

[40] G. Strang, Wavelets, American Scientist JSTOR 82(3) (1994) 250–255.

[41] C. V. Loan, Computational Frameworks for the Fast Fourier Transform, SIAM Publications, Philadelphia, USA, 1992.

[42] J. Dongarra, F. Sullivan, Guest editors introduction to the top 10 algorithms, Computing in Science Engineering 2(1) (2000) 22–23.

[43] C. Burrus, M. Frigo, S. Johnson, M. Pueschel, I. Selesnick, Fast Fourier Transforms, Samurai Media Limited, 2018.
URL `https://www.amazon.ca/Fast-Fourier-Transforms-Sidney-Burrus/dp/988840752X`

[44] C. Burrus, M. Frigo, S. Johnson, M. Pueschel, I. Selesnick, Fast Fourier Transforms (6x9 Version), Connexions, 2012.
URL `https://cnx.org/contents/Fujl6E8i@5.8:gGcNzVsy@10/Introduction-Fast-Fourier-Transforms`

[45] C. M. Rader, Discrete fourier transforms when the number of data samples is prime, Proc. IEEE 56 (1968) 1107–1108.

[46] M. Frigo, S. G. Johnson, The design and implementation of fftw3, Proceedings of the IEEE 93 (2005) 216–231.

[47] L. Bluestein, A linear filtering approach to the computation of discrete fourier transform, IEEE Transactions on Audio and Electroacoustics 18 (4) (1970) 451–455. `doi:10.1109/TAU.1970.1162132`.

[48] S. Winograd, On computing the discrete fourier transform, Proceedings of the National Academy of Sciences 73 (4) (1976) 1005–1006. `doi:10.1073/pnas.73.4.1005`.
URL `https://www.pnas.org/doi/abs/10.1073/pnas.73.4.1005`