Strategies for the Synthesis of Fast Algorithms for the Computation of the Matrix-vector Products

Aleksandr Cariow

Received 16 June 2014; Published online 4 October 2014

© The author(s) 2014. Published with open access at www.uscip.us

Abstract

This article offers the strategies for the synthesis of fast algorithms for computing the matrix-vector products. It considers the specific example of synthesis of fast algorithm for matrix by the vector multiplication. The example offered allows tracking all the stages of construction of the algorithm which was rationalized from the point of view of number multiplication minimization.

Keywords: Matrix-vector multiplication; Fast algorithms; Reduction of the number of arithmetic operations

1. Introduction

The matrix-vector multiplication is one of the most important and basic operations in science and engineering. The necessity of computing a matrix-vector product arises in many numeric tasks that are connected to the digital signal processing (DSP), image detection, audio and video compression (Blahut, 1985). The operations typical for digital signal processing, such as the linear and circular convolution, correlation, discrete orthogonal transforms in various bases, multiresolution decomposition and reconstruction are also reduced to the matrix-vector multiplication. Through the analysis of different basic matrix structures it was discovered that some interrelations between their elements allow the reduction of the number of arithmetic operations. This fact gave rise to the works that concern the synthesis of algorithms which are connected with the realization of matrix-vector products, optimized in the aspect of the number of multiplication and addition operations. Fast Fourier transform is an example of an algorithm created with the use of favorable relations between the elements inside a matrix. In order to reduce the number of arithmetic operations necessary for its realization, this algorithm exploits the features of symmetry and periodicity of the discrete exponential functions that are the elements of a discreet Fourier transform base matrix (Blahut, 1985). The other „fast” algorithms for the calculation of the discreet Fourier transform and
“fast convolution”, such as the one presented by S. Winograd on the basis of the Chinese remainder theorem and the Cook-Toom algorithm, which calculated a linear convolution on the basis of the effective polynomial multiplication, which is, in turn, based on Lagrange polynomial interpolation, appeared (Nussbaumer, 1982). Furthermore several modifications of the “fast orthogonal transforms” algorithms appeared, for instance: DCT (discrete cosine transform), DST (discrete sine transform), DHT (discrete Hartley transform), Slant transform and others, described in the works of R. C. Agarwal, N. Ahmed, C.S. Burrus, D. P. Kolba, T.W. Parks, C.M. Rader, K.R. Rao, J.H. McClellan, H.J. Nussbaumer and R.N. Bracewell (Nussbaumer, 1982; Burrus et al., 1999; Tolimieri et al., 1989; Moon and Stirling., 2000).

It should be emphasized that the mathematical notations used by the authors to prove their ideas for creating the fast algorithms are often quite complicated (the polynomial theory, number theory, Galois field theory etc). As a result the practitioners unfamiliar with these issues may find it difficult to understand the essence of the suggested solutions. Moreover, the theories mentioned and the invented algorithms concern a narrow circle of specialized tasks. Apart from the enumerated solutions there exist many other tasks in which the structures of matrices that allow the reduction of computing operations can be found. Unfortunately, there is no universal and accessible method that would allow the synthesis of the fast algorithms. However, the existence of an incredibly simple and effective methodology (or actually a set of strategies) for the search of algorithmic solutions for the synthesis of optimized (or at least rationalized) algorithms for the matrix-vector multiplication from the standpoint of the number of necessary arithmetic operations was revealed. These strategies will be presented in the further part of this article.

2. Defining the Set of Matrix Pattern Structures that Allow the Reduction of the Number of Arithmetic Operations while Computing Matrix-vector Product

The aim of computing rationalization while computing matrix-vector products is to minimize the total number of multiplication and addition operations (Blahut, 1985). It should be emphasized that it is not always possible to reduce the number of additions and multiplications. Sometimes it is possible to reduce the number of multiplication operations at the cost of increasing the number of additions. However, the total number of operations often remains lower than the original. In such cases the computational “gain” is obvious. On the other hand, if as a result of growth additions the total number of arithmetic operations is bigger than the original, the gain of such a rationalization needs to be justified.

Thus, the strategies of the reduction of arithmetic operations in calculating the matrix-vector products will be presented.

Table 1 presents an incomplete set of typical 2×2 matrix structures (they shall be called “patterns” or “pattern matrices”), the block allocation of which allows to reduce the number of arithmetic operations in calculating matrix-vector products. The possibility of the reduction of operations for some of these structures has been, among others, presented in the work (Țariov and Adamski, 2005) for 2×2 matrices and (Țariov and Adamski, 2006) for N×N matrices. The extended set of computational procedures for 2×2 matrices, which accounts for such a reduction, has been
presented in Table 2. Table 3 presents a set of typical N×N matrix structures, the block constellation of which allows reducing the number of arithmetic operations in calculating matrix-vector products.

**Table 1** The specific structures of 2×2 matrix patterns

<table>
<thead>
<tr>
<th>Structural constellations of 2×2 matrices</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_2^{(1)} = \begin{bmatrix} a &amp; b \ c &amp; d \end{bmatrix}$</td>
</tr>
<tr>
<td>$A_2^{(5)} = \begin{bmatrix} a &amp; b \ c &amp; -a \end{bmatrix}$</td>
</tr>
<tr>
<td>$A_2^{(9)} = \begin{bmatrix} a &amp; b \ b &amp; b \end{bmatrix}$</td>
</tr>
<tr>
<td>$A_2^{(13)} = \begin{bmatrix} a &amp; a \ b &amp; b \end{bmatrix}$</td>
</tr>
</tbody>
</table>

**Table 2** Matrix-vector computational procedures for $A_i^{(i)}$ matrix patterns, $i = 1, 2, \ldots, 16$.

<table>
<thead>
<tr>
<th>No</th>
<th>Matrix-vector computing procedures</th>
<th>No</th>
<th>Matrix-vector computing procedures</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td>$Y_{2×1}^{(1)} = T_{2×2}^{(2)} \mathbf{diag} (a, c, b, d) T_{2×2}^{(2)} X_{2×1}$</td>
<td>9</td>
<td>$Y_{2×1}^{(9)} = T_{2×2}^{(2)} \mathbf{diag} (s, \oplus b) T_{2×2}^{(2)} X_{2×1}$</td>
</tr>
<tr>
<td>1b</td>
<td>$Y_{2×1}^{(10)} = T_{2×2}^{(2)} \mathbf{diag} (a, b, c, d) T_{2×2}^{(2)} X_{2×1}$</td>
<td>10</td>
<td>$Y_{2×1}^{(10)} = T_{2×2}^{(2)} \mathbf{diag} (s, \oplus b) T_{2×2}^{(2)} X_{2×1}$</td>
</tr>
<tr>
<td>2</td>
<td>$Y_{2×1}^{(11)} = T_{2×2}^{(2)} \mathbf{diag} (a, c, b) T_{2×2}^{(2)} X_{2×1}$</td>
<td>11</td>
<td>$Y_{2×1}^{(11)} = (a \oplus b) \mathbf{H}<em>2 X</em>{2×1}$</td>
</tr>
<tr>
<td>3</td>
<td>$Y_{2×1}^{(12)} = T_{2×2}^{(2)} \mathbf{diag} ((c-a), (b-a), a) T_{2×2}^{(2)} X_{2×1}$</td>
<td>12</td>
<td>$Y_{2×1}^{(12)} = (a \oplus b) \mathbf{H}<em>2 X</em>{2×1}$</td>
</tr>
<tr>
<td>4</td>
<td>$Y_{2×1}^{(13)} = T_{2×2}^{(2)} \mathbf{diag} ((c-a), (b-a), (a)) T_{2×2}^{(2)} X_{2×1}$</td>
<td>13</td>
<td>$Y_{2×1}^{(13)} = (a \oplus b) \mathbf{I}<em>{2×2} \mathbf{I}</em>{2×2} X_{2×1}$</td>
</tr>
<tr>
<td>5</td>
<td>$Y_{2×1}^{(14)} = T_{2×2}^{(2)} \mathbf{diag} ((a-b), -(a-b), b) T_{2×2}^{(2)} X_{2×1}$</td>
<td>14</td>
<td>$Y_{2×1}^{(14)} = (a \oplus b) \mathbf{I}<em>{2×2} \mathbf{I}</em>{2×2} X_{2×1}$</td>
</tr>
<tr>
<td>6</td>
<td>$Y_{2×1}^{(15)} = T_{2×2}^{(2)} \mathbf{diag} ((a-a), -b, b) T_{2×2}^{(2)} X_{2×1}$</td>
<td>15</td>
<td>$Y_{2×1}^{(15)} = (a \oplus b) \mathbf{I}<em>{2×2} \mathbf{I}</em>{2×2} X_{2×1}$</td>
</tr>
<tr>
<td>7</td>
<td>$Y_{2×1}^{(16)} = (a \oplus b) \odot (a-b) \mathbf{H}<em>2 X</em>{2×1}$</td>
<td>16</td>
<td>$Y_{2×1}^{(16)} = (a \oplus b) \mathbf{I}<em>{2×2} \mathbf{I}</em>{2×2} X_{2×1}$</td>
</tr>
</tbody>
</table>

where $\mathbf{I}_{M×N}$ - is a $M \times N$ matrix of ones (a matrix where every element is equal to one), Matrix $\mathbf{H}_2 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ is a $2 \times 2$ Hadamard matrix, $\mathbf{I}_{N}$ - is an identity $N \times N$ matrix and signs “\(\odot\)”, “\(\otimes\)” denote tensor product and a direct sum of two matrices, respectively (Regalia, P. A. and Mitra, S. K., 1989),

\[
\mathbf{I}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{I}_{2×1} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \mathbf{I}_{2×2} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \mathbf{T}_{2×2}^{(1)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{T}_{2×2}^{(2)} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \quad \mathbf{T}_{2×3}^{(1)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{T}_{2×3}^{(2)} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix}.
\]
ed matrices denote the operations of data transfer. Points where elements of the result vector (out remember the row or column of the original matrix is possible before permuting the rows and/or columns. However, if the block structure of the matrix corresponds to the rows of a matrix makes it necessary to shuffle the elements of the result vector (output), while permuting the columns, which will require the prior permutation of the input vector elements.

Figures 1-16 represent data flow diagrams for the 2×2 matrix-vector multiplication for the matrix patterns 1-16, respectively. Figures 17-32 show the data flow diagrams for the N×N matrix-vector multiplication for the matrix patterns 17-32. In this paper, the data flow diagrams are oriented from left to right. Straight lines in the figures denote the operations of data transfer. Points where lines converge represent summation. The dashed lines indicate the sign change operation. I have deliberately used the usual lines without arrows, so as not to clutter the picture. The circles in these figures show the operation of multiplication by a variable (or constant) which is inscribed inside a circle. The rectangles indicate the multiplication blocks of matrices inscribed inside them by the suitable subvectors of data.

3. The Strategies of the Construction of “Fast” Matrix-vector Multiplication algorithms

First of the strategies for the rationalization of computation while computing matrix-vector products depends on finding such fragments of a matrix structure, which allocation would agree with any of the pattern structures. Fortunately, the matrices that occur in the digital signal processing have such a specific block structures or may be brought to such form by rearranging their rows or columns.

Firstly, we should divide in mind the original matrix into the four submatrices. Then, the structure of the new blocks, which were formed through that division, should be matched with the structure of block of the one of patterns (of pattern matrices from Table 3). In this way, one of the computing procedures from (1) to (16) is chosen. If the block structure that we have found matches the patterns 2-16, it means that we have managed to minimize the number of arithmetic operations. Yet, if the block structure of the matrix corresponds to the pattern 1, it means that we cannot rationalize the multiplication by such a matrix.

In such a case the permutation of the rows and/or columns of the original matrix is possible before we retry to match the pattern. A success in finding a suitable block structure depends largely on the fortunate choice of permutation of the rows and/or columns. However, it should be remembered, that the permutation of the rows of a matrix makes it necessary to shuffle the elements of the result vector (output), while permuting the columns, which will require the prior permutation of the input vector elements.
Also, it needs to be underlined that initially the rationalized procedure of vector and matrix multiplication cannot be found, the conclusions cannot be drawn and it is also not possible in the next steps. Hence, regardless of the result of the first step, the second step should be taken. The next step of the rationalization of the algorithm of multiplying a vector by a matrix is the analogical division of each of the three (or four in the worst case) submatrices. These matrices emerged as a result of the synthesis of the computing procedure in the previous step, and of an attempt to find an adequate block structure for each of those submatrices, that matches the pattern. Then, an adequate procedure for each of these submatrices should be chosen and, on the base of it, a common overall procedure for this step of rationalization should be created. In case of finding the matching pattern (from numbers 1 to 10) for our block structure, at least for one of the submatrices, we obtain even more rationalized total computing procedure.

Fig. 1a.  
\[ \varphi_0 = c - a, \quad \varphi_1 = b - a \]

Fig. 1b.  
\[ s_0 = a + b, \quad s_0 = a + b \]

Fig. 2.  
\[ s_0 = a + b, \quad s_1 = a - b \]

Fig. 3.  
\[ \tilde{s}_0 = (a + b)/2, \quad \tilde{s}_1 = (a - b)/2 \]

Fig. 4.  
\[ \tilde{s}_0 = (a + b)/2, \quad \tilde{s}_1 = (a - b)/2 \]

Fig. 5.  
\[ \tilde{s}_0 = (a + b)/2, \quad \tilde{s}_1 = (a - b)/2 \]

Fig. 6.  
\[ \tilde{s}_0 = (a + b)/2, \quad \tilde{s}_1 = (a - b)/2 \]

Fig. 7.  
\[ \tilde{s}_0 = (a + b)/2, \quad \tilde{s}_1 = (a - b)/2 \]

Fig. 8.  
\[ \tilde{s}_0 = (a + b)/2, \quad \tilde{s}_1 = (a - b)/2 \]

Fig. 9.  
\[ \tilde{s}_0 = (a + b)/2, \quad \tilde{s}_1 = (a - b)/2 \]

Fig. 10.  
\[ \tilde{s}_0 = (a + b)/2, \quad \tilde{s}_1 = (a - b)/2 \]

Fig. 11.  
\[ \tilde{s}_0 = (a + b)/2, \quad \tilde{s}_1 = (a - b)/2 \]

Fig. 12.  
\[ \tilde{s}_0 = (a + b)/2, \quad \tilde{s}_1 = (a - b)/2 \]

Fig. 13.  
\[ \tilde{s}_0 = (a + b)/2, \quad \tilde{s}_1 = (a - b)/2 \]

Fig. 14.  
\[ \tilde{s}_0 = (a + b)/2, \quad \tilde{s}_1 = (a - b)/2 \]

Fig. 15.  
\[ \tilde{s}_0 = (a + b)/2, \quad \tilde{s}_1 = (a - b)/2 \]

Fig. 16.  
\[ \tilde{s}_0 = (a + b)/2, \quad \tilde{s}_1 = (a - b)/2 \]
Following this procedure, it is possible to get a submatrix of the 2×2 order, and to reduce the number of necessary arithmetic operations to the minimum. However, such a complete decomposition cannot always be achieved, especially not for all the submatrices which were generated as a result of block division of the original matrices and not at each step of decomposition. On the one hand, it depends on the structural properties of a matrix, and on the other, on the designer’s invention – the wrong choice of the rows or columns of a matrix (or submatrix) to shuffle at each step may lead to a non-optimal decomposition of the original matrices.

It also needs to be emphasized that an iterating division of original matrices into four submatrices may not always lead to a reduced number of operations, even after the several permutation of rows and/or columns of those matrices. In such case it is worth to search for other alternatives. For example, the original matrix can be presented in a form of a sum or a subtraction result of two matrices and then the above described ways of rationalization can be applied. Naturally, such a transformation is sensible, when the total number of operations after the decomposition is lower than the original.

Another approach may be to try to find the single fragments (subblocks) of a matrix that have a pattern structure. In the latter case the reduction of arithmetic operations may be achieved only for the selected fragments.

**Table 3** The specific structures of N×N matrix patterns

<table>
<thead>
<tr>
<th>Pattern type</th>
<th>Number of arithmetical operations: «×»; «+»</th>
<th>Pattern type</th>
<th>Number of arithmetical operations: «×»; «+»</th>
</tr>
</thead>
<tbody>
<tr>
<td>A(1) N</td>
<td>[ \begin{bmatrix} \alpha_{N/2} &amp; \beta_{N/2} \ \gamma_{N/2} &amp; \delta_{N/2} \end{bmatrix} ]</td>
<td>N²; N(N−1)</td>
<td>A(2) N</td>
</tr>
<tr>
<td>A(3) N</td>
<td>[ \begin{bmatrix} \alpha_{N/2} &amp; \beta_{N/2} \ \gamma_{N/2} &amp; \delta_{N/2} \end{bmatrix} ]</td>
<td>3N²/4; N(3N−4)/4</td>
<td>A(4) N</td>
</tr>
<tr>
<td>A(5) N</td>
<td>[ \begin{bmatrix} \alpha_{N/2} &amp; \beta_{N/2} \ \gamma_{N/2} &amp; \delta_{N/2} \end{bmatrix} ]</td>
<td>3N²/4; 5N²/4</td>
<td>A(6) N</td>
</tr>
<tr>
<td>A(7) N</td>
<td>[ \begin{bmatrix} \alpha_{N/2} &amp; \beta_{N/2} \ \gamma_{N/2} &amp; \delta_{N/2} \end{bmatrix} ]</td>
<td>N²/2; N(N+1)</td>
<td>A(8) N</td>
</tr>
<tr>
<td>A(9) N</td>
<td>[ \begin{bmatrix} \alpha_{N/2} &amp; \beta_{N/2} \ \gamma_{N/2} &amp; \delta_{N/2} \end{bmatrix} ]</td>
<td>N²/2; 3N²/4</td>
<td>A(10) N</td>
</tr>
<tr>
<td>A(11) N</td>
<td>[ \begin{bmatrix} \alpha_{N/2} &amp; \beta_{N/2} \ \gamma_{N/2} &amp; \delta_{N/2} \end{bmatrix} ]</td>
<td>N²/2; N²/2</td>
<td>A(12) N</td>
</tr>
</tbody>
</table>
\[ A_{N}^{(13)} = \begin{bmatrix} A_{N/2} & \vdots & A_{N/2} \\ B_{N/2} & \vdots & B_{N/2} \end{bmatrix} \quad N^2/4; \quad N(N-1)/2 \]

\[ A_{N}^{(15)} = \begin{bmatrix} A_{N/2} & \vdots & A_{N/2} \\ B_{N/2} & \vdots & B_{N/2} \end{bmatrix} \quad N^2/4, \quad N^2/4 \]

\[ A_{N}^{(14)} = \begin{bmatrix} A_{N/2} & \vdots & B_{N/2} \\ A_{N/2} & \vdots & B_{N/2} \end{bmatrix} \quad N^2/4; \quad N(N-1)/2 \]

\[ A_{N}^{(16)} = \begin{bmatrix} -A_{N/2} & \vdots & A_{N/2} \\ B_{N/2} & \vdots & -A_{N/2} \end{bmatrix} \quad N^2/4, \quad N^2/4 \]

\[ Y_{N=4}^{(1a)} = (T_{2x2}^{(1)} \otimes I_{N/2}) (A_{N/2} + C_{N/2} + B_{N/2} + D_{N/2}) (T_{2x2}^{(2)} \otimes I_{N/2}) X_{N=4} \]

\[ Y_{N=4}^{(1b)} = (T_{2x2}^{(1)} \otimes I_{N/2}) (A_{N/2} \circ B_{N/2} \circ C_{N/2} \circ D_{N/2}) (T_{2x2}^{(2)} \otimes I_{N/2}) X_{N=4} \]

\[ Y_{N=4}^{(1c)} = (T_{2x2}^{(1)} \otimes I_{N/2}) (A_{N/2} \circ B_{N/2} \circ C_{N/2} \circ D_{N/2}) (T_{2x2}^{(2)} \otimes I_{N/2}) X_{N=4} \]

\[ Y_{N=4}^{(1d)} = (T_{2x2}^{(1)} \otimes I_{N/2}) (A_{N/2} \circ B_{N/2} \circ C_{N/2} \circ D_{N/2}) (T_{2x2}^{(2)} \otimes I_{N/2}) X_{N=4} \]

\[ Y_{N=4}^{(1e)} = (T_{2x2}^{(1)} \otimes I_{N/2}) (A_{N/2} \circ B_{N/2} \circ C_{N/2} \circ D_{N/2}) (T_{2x2}^{(2)} \otimes I_{N/2}) X_{N=4} \]

\[ Y_{N=4}^{(1f)} = (T_{2x2}^{(1)} \otimes I_{N/2}) (A_{N/2} \circ B_{N/2} \circ C_{N/2} \circ D_{N/2}) (T_{2x2}^{(2)} \otimes I_{N/2}) X_{N=4} \]

\[ Y_{N=4}^{(1g)} = (T_{2x2}^{(1)} \otimes I_{N/2}) (A_{N/2} \circ B_{N/2} \circ C_{N/2} \circ D_{N/2}) (T_{2x2}^{(2)} \otimes I_{N/2}) X_{N=4} \]

\[ Y_{N=4}^{(1h)} = (T_{2x2}^{(1)} \otimes I_{N/2}) (A_{N/2} \circ B_{N/2} \circ C_{N/2} \circ D_{N/2}) (T_{2x2}^{(2)} \otimes I_{N/2}) X_{N=4} \]

\[ Y_{N=4}^{(1i)} = (T_{2x2}^{(1)} \otimes I_{N/2}) (A_{N/2} \circ B_{N/2} \circ C_{N/2} \circ D_{N/2}) (T_{2x2}^{(2)} \otimes I_{N/2}) X_{N=4} \]

\[ Y_{N=4}^{(1j)} = (T_{2x2}^{(1)} \otimes I_{N/2}) (A_{N/2} \circ B_{N/2} \circ C_{N/2} \circ D_{N/2}) (T_{2x2}^{(2)} \otimes I_{N/2}) X_{N=4} \]

\[ Y_{N=4}^{(1k)} = (T_{2x2}^{(1)} \otimes I_{N/2}) (A_{N/2} \circ B_{N/2} \circ C_{N/2} \circ D_{N/2}) (T_{2x2}^{(2)} \otimes I_{N/2}) X_{N=4} \]
The technique of designing fast algorithms for the vector-matrix multiplication based on the presented considerations is shown schematically in Figure 33.
Fig. 20.

Fig. 21.

Fig. 22.

Fig. 23.

Fig. 24.
The strategies of computing rationalization in the synthesis of fast algorithms for the matrix-vector multiplication


Let us consider a synthesis of a fast algorithm for calculating a matrix-vector product for a matrix of arbitrary values. Suppose that the matrix has the following form:
The task is to obtain a product of the multiplication of the above matrix by a vector of data $X_{8\times 1}$ thus, a product is:

$$Y_{8\times 1} = A_8 X_{8\times 1}$$  \hspace{1cm} (33)

It should be noticed that, in case of the realization of the multiplication procedure in a direct way, which is according to (33), it would be necessary to apply 64 multiplications and 56 additions. Let us try to rationalize the computation process in order to reduce the number of multiplications at least by a half. It can be observed that the matrix $A_8$ has a specific block structure:

$$A_8 = \begin{bmatrix} A_4 & B_4 \\ B_4 & A_4 \end{bmatrix}$$

The structure of blocks corresponds to the pattern structure 7 which was given in Table 3. Then, there exists a possibility to reduce the number of arithmetic operations. Hence, the rationalized procedure of defining the product considered will take the following form:

$$Y_{8\times 1} = (H_2 \otimes I_4)D_8^{(i)}(H_2 \otimes I_4)X_{8\times 1},$$  \hspace{1cm} (34)

where

$$D_8^{(i)} = \frac{1}{2}(A_4 + B_4) \oplus \frac{1}{2}(A_4 - B_4);$$

$$H_2 \otimes I_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix},$$

and


\[ A_4 + B_4 = \begin{bmatrix} a + f & b + g & d + h & e + k \\ c + g & a + f & e + k & d + h \\ d + h & e + k & a + f & b + g \\ e + k & d + h & c + g & a + f \end{bmatrix}, \quad A_4 - B_4 = \begin{bmatrix} a - f & b - g & d - h & e - k \\ c - g & a - f & e - k & d - h \\ d - h & e - k & a - f & b - g \\ e - k & d - h & c - g & a - f \end{bmatrix} \] (35)

Figure 33 presents a data flow diagram which represents the algorithmic structure of the computation process for the multiplication of the matrix \( A_n \) by vector \( X_{8x1} \) at the first decomposition level.

It should be observed that the submatrices (35) that emerged as a result of procedure (34) have also the same block constellation as that of matrix \( A_n \):

\[ A_4 + B_4 = \begin{bmatrix} A_1 & B_1 \\ B_2 & A_2 \end{bmatrix}, \quad A_4 - B_4 = \begin{bmatrix} C_1 & D_1 \\ D_2 & C_2 \end{bmatrix}, \]

where:

\[ A_2 = \begin{bmatrix} a + f & b + g \\ c + g & a + f \end{bmatrix}, \quad B_2 = \begin{bmatrix} d + h & e + k \\ e + k & d + h \end{bmatrix}, \quad C_2 = \begin{bmatrix} a - f & b - g \\ c - g & a - f \end{bmatrix}, \quad D_2 = \begin{bmatrix} d - h & e - k \\ e - k & d - h \end{bmatrix}. \]

Fig. 34. Data flow diagram showing the organization of a computing process for computing a product of the matrix \( A_n \) by vector \( X_{8x1} \) after the first step of rationalization.

Thus, we can once again apply the same method for the reduction of the number of operations for each of the created submatrices.

Through the application of an adequate decomposition of matrix components that was demonstrated in the procedure (34) and of the model presented in the Figure 34, we synthesize a new, specified computing procedure for the second step of the algorithm rationalization. Then, the procedure for calculation of the matrix-vector product will take the following form:

\[ Y_{8x1} = (H_2 \otimes I_2)(I_2 \otimes H_2 \otimes I_2)D_8^{(2)}(I_2 \otimes H_2 \otimes I_2)(H_2 \otimes I_2)X_{8x1} \] (36)
D^{(2)}_8 = \text{diag} \left[ \frac{1}{4} (A_2 + B_2), \frac{1}{4} (A_2 - B_2), \frac{1}{4} (C_2 + D_2), \frac{1}{4} (C_2 - D_2) \right], \quad I_2 \otimes H_2 \otimes I_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 1 \\ 1 & -1 \end{bmatrix}.

\begin{align*}
A_2 + B_2 &= \begin{bmatrix} a + f + d + h & b + g + e + k \\ c + g + e + k & a + f + d + h \end{bmatrix}, \\
A_2 - B_2 &= \begin{bmatrix} a + f - d - h & b + g - e - k \\ c + g - e - k & a + f - d - h \end{bmatrix}, \\
C_2 + D_2 &= \begin{bmatrix} a - f + d + h & b - g + e + k \\ c - g + e + k & a - f + d + h \end{bmatrix}, \\
C_2 - D_2 &= \begin{bmatrix} a - f - d - h & b - g - e - k \\ c - g - e - k & a - f - d - h \end{bmatrix}.
\end{align*}

The data flow diagram that reflects the algorithm structure of the computing process on this stage of rationalization has been presented in Figure 35.

Now one may notice that the matrices \(A_2 + B_2, A_2 - B_2, C_2 + D_2, C_2 - D_2\), which constitute the matrix \(D^{(2)}_8\) of procedure (36), have the block structure which is compatible with structure 4 from Table 3. In such case the determination of the products of the appropriate subvectors and the marked matrices may also be rationalized through the use of procedure (4).

\[ Y_{x_{i+1}} = W^{(2)}_iy_{x_{i}}W^{(1)}_{T_{i+1}T_{i}}D^{(2)}_8(T_{12}T_{12})W^{(1)}_yW^{(2)}_xX_{x_{i+1}} \]

(37)

Fig. 35. Data flow diagram for the organization of a computing process for the computation of a product of matrix \(A_8\) by the vector \(X_{x_{i+1}}\) after the second stage of rationalization.

As a result we obtain:
where:

$$D^{(3)}_{12} = \frac{1}{4} \text{diag}(s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}, s_{11}),$$

$$W^{(1)}_8 = (I_2 \otimes H_2 \otimes I_2), \quad W^{(2)}_8 = (H_2 \otimes I_4), \quad \tilde{T}^{(3)}_{8 \times 12} = (I_4 \otimes T^{(3)}_{3 \times 2}), \quad \tilde{T}^{(3)}_{12 \times 8} = (I_4 \otimes T^{(3)}_{3 \times 2})$$

while:

\[
\begin{align*}
    s_0 &= c + g + e + k - a - f - d - h, \quad s_1 = b + g + e + k - a - f - d - h, \quad s_2 = a + f + d + h, \\
    s_3 &= c + g - e - k - a - f + d + h, \quad s_4 = b + g - e - k - a - f + d + h, \quad s_5 = a + f - d - h, \\
    s_6 &= c - g + e - k - a + f - d + h, \quad s_7 = b - g + e - k - a + f - d + h, \quad s_8 = a - f + d - h, \\
    s_9 &= c - g - e + k - a + f + d + h, \quad s_{10} = b - g - e + k - a + f + d + h, \quad s_{11} = a - f - d + h.
\end{align*}
\]  

(38)

In Figure 36 a data flow diagram which represents the final version of rationalized computing process of calculating the product of the matrix $A_8$ and vector $X_{8\times 1}$ has been presented. Evidently, the synthesized algorithm for the multiplication of a matrix for the example considered requires only 12 multiplications instead of 64, and 44 addition operations instead of 56.

**Fig 36.** Data flow diagram of the organization of a computing process for computing a product of matrix $A_8$ by vector $X_{8\times 1}$ after the third stage of rationalization

As it can be observed, in the example analyzed, the synthesized algorithm of the matrix multiplication by the vector requires only 12 multiplications, instead of 64, and 112 operations of addition, instead of 56. The joint number of arithmetic operations for the rationalized algorithm amounts to 124 operations, whereas for the original algorithm (non-rationalized), this number amounts to 120. However, when we consider the multiplication of the vector by a constant matrix
(in which the elements are constants), and when the elements \( s_i \), \( i = 0,1,...,11 \) might be determined in advance, the number of the additions is lowered to 44. In such case, the joint number of arithmetic operations will amount to 56, so the number will be reduced by half.

The repetitions of certain calculations and relations which could be taken into account in the further rationalization of the calculation process can be observed thanks to the analysis of the system of relations, used to assign the elements \( s_i \). By skipping the transitional transformations, the procedure for the value calculation \( \{s_i\} \) can be presented in the following way:

\[
S_{12:1} = \tilde{P}_{12} \tilde{\Sigma}_{12} (I_6 \otimes H_2) R_{12:10}^{(2)} (I_5 \otimes H_2) R_{10:9}^{(1)} A_{9:1},
\]

where \( S_{12:1} = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}, s_{11}\} \), and \( A_{9:1} = \{a, b, c, d, g, e, f, h, k\} \).
The graph-structural model which demonstrates the procedure (39) was presented in the figure 37. As it can be seen, in case of the further rationalization, the number of arithmetic operations for the realization of the analyzed product amounts to: 12 multiplications and, generally, to 74 additions, which totals 86 operations. This number of operations is almost 40% lower than in case of applying the direct method of calculation.

**Fig 37.** The data flow diagram describing the process of calculation of elements of the vector $S_{12:1}$, in accordance with the procedure (39).
5. Summary

The strategies of finding rational solutions during the synthesis of the fast algorithms of multiplying a vector by a matrix that were presented in this article, constitute a useful instrument for designing the effective digital signal processing algorithms. It is a relatively simple and effective approach which allows to reduce the number of arithmetic operations during the matrix-vector product calculation, which bases on the specific structure of the matrices. The suggested approach can be used during the synthesis of optimized algorithms for the tasks which are not included in the so far existing solutions. It should also be emphasized that the suggested approach allows to repeat (duplicate) the results, or to synthesize the completely new fast algorithms (of different algorithmic structure) for the typical digital signal processing tasks that are already solved and presented in the scientific literature. Articles (Ţariov, 2012.), (Cariow and Cariowa, 2012), (Majorkowska-Mech and Cariow, 2012), (Cariow and Gliszczynski, 2012), (Cariow and Cariowa, 2013), (Cariow and Cariowa, 2014), (Cariow and Majorkowska-Mech, 2014), show the application of the described methodology for the synthesis of some fast data processing algorithms.

References

\url{http://dx.doi.org/10.1002/0470097434}
Siam Review, 31, No. 4, 586-613.  
\url{http://dx.doi.org/10.1137/1031127}