Parallelization of Spline-Wavelet Decomposition

YURI K.DEM'YANOVICH
St. Petersburg State University
7/9 Universitetskaya nab., St. Petersburg
RUSSIA
y.demjanovich@spbu.ru

Abstract: A discrete spline-wavelet decomposition of the first order is discussed in the framework of the non-classical approach. The purpose of this paper is to estimate the calculation duration for the discrete spline-wavelet decomposition with the use of two sorts of computers: One-Processor System (OPS) and Parallel Multi-processor System (PMS). The main object is the grid functions, which are named flows. The finite dimensional spaces of the initial flows, wavelet flows and main flows are introduced. These spaces are associated with the original and the enlarged grids, respectively. Estimates for the duration of the calculations are given with taking into account the properties of a communication computer environment. The presentation is accompanied with illustrative examples. We consider the grid functions whose domain is a grid on the real axis (for example, on the set of integers). This approach is convenient when processing flows are sequences of numbers. Then we discuss a grid enlargement and construct an embedded discrete spline space. Using a projection operator, we obtain a wavelet decomposition and give an illustration example of the mentioned decomposition. Taking into account the obtained algorithms we consider their implementation with OPS and PMS. In the situation of the unlimited concurrency the duration (runtime) of calculation with PMS does not depend on the data volume (i.e. it does not depend on the length of the initial flow), on the other hand, the duration of the calculation with OPS is directly proportional to the data volume.

Key–Words: spline-wavelet decomposition, parallelization, runtime, duration of calculation

1 Introduction

Wavelet expansions are widely used in processing numerical information flows. The volumes of such flows are constantly increasing. It is a stimulus to the further development of the theory of wavelets (see [1]-[2]). The approach to constructing wavelets used in this paper is based on the approximation relations (see, for example, [3]).

In contrast to the classic wavelets (see [1]), the mentioned approach allows us to use an irregular grid (both finite and infinite). The last one is very important for saving the computer resources in the case of singular changes of data flow. The building wavelet decompositions in the multidimensional case and on an arbitrary differentiable manifold (see [4]) can be applied to the finite element approximations (for example, see [7] – [21]). In this way the possibilities of wavelet decomposition are significantly expanded.

The construction of the wavelet basis in different functional spaces is very difficult in the classical cases. The discussed approach does not require the preliminary construction of the wavelet basis (if someone wishes to construct this basis they can obtain it in the future). On the other hand, the knowledge of the wavelet basis allows us to achieve substantial savings of computer and network resources.

Note that the mentioned savings do not need a wavelet basis in spaces of functions with a continual domain. It suffices to get a suitable basis for the space of wavelet numerical flows, but for this it is necessary to perform all constructions without use of functions with a continual domain (see [5] – [6]).

The purpose of this paper is to estimate the calculation duration (runtime) for the discrete spline-wavelet decomposition with the use of two sorts of computers: One Processor System (OPS) and Parallel Multi-processor System (PMS). We discuss computation complexity of parallelization for the decomposition and reconstruction algorithms. At first we consider their implementation on OPS, and then — on PMS.

In the situation of the unlimited concurrency the duration of calculation with PMS does not depend on the data volume (i.e. it does not depend on the length of the initial flow), on the other hand, the duration of calculation with OPS is directly proportional to the data volume.

2 Background information

In this paper, we consider the grid functions whose domain is a grid on the real axis (for example, on the set of integers). This approach is convenient when processing flows are sequences of numbers.

Consider the grid on the real axis,

$$\Xi : \quad \ldots < \xi_{-2} < \xi_{-1} < \xi_0 < \xi_1 < \xi_2 \ldots .$$

The set of functions $u(t)$ defined on the grid Ξ is denoted by $C(\Xi)$. It is clear that $C(\Xi)$ is a linear space.

If $a \in \Xi$, then there is $i \in \mathcal{Z}$ such that $a = \xi_i$. In this case, we denote $a^- = \xi_{i-1}$, $a^+ = \xi_{i+1}$. Let's assume that

$$a, b \in \Xi, \qquad a^+ < b^-,$$

i.e. for some $i, j \in \mathcal{Z}$, $i + 2 < j$, the equalities are $a = \xi_i$, $b = \xi_j$. For mentioned a and b we introduce the notation $\|a, b\| = \{\xi_s \mid a \le \xi_s \le b, \ s \in \mathcal{Z}\}$. The set $\|a, b\|$ will be called *a grid segment*.

Consider a linear space $C\|a, b\|$ of functions $u(t)$ defined on the grid segment $\|a, b\|$. Obviously, the space $C\|a, b\|$ has a finite dimension.

3 Grid enlargement

For a natural number m we denote

$$J_m = \{0, 1, \ldots, m\}, \qquad J'_m = \{-1, 0, 1, \ldots, m\}.$$

On the grid segment $\|a, b\|$,

$$a = \xi_0 < \xi_1 < \ldots < \xi_{M-1} < \xi_M = b,$$

consider the functions $\{\omega_j(t)\}_{j \in J'_{M-1}}$ as elements of the space $C\|a, b\|$

$$\omega_j(\xi_s) = \delta_{s,j+1}, \qquad s \in J_M. \tag{1}$$

Below, we assume that if $c > d$ then the set $\|c, d\|$ is empty.

Let $5 \le K < M$. Consider the injective mapping κ of the set J_K into the set J_M, for which

$$\kappa(0) = 0, \quad \kappa(i) < \kappa(i + 1), \quad \kappa(K) = M.$$

We introduce the set $J^* \subset J_M$ given by the formula

$$J^* = \kappa J_K. \tag{2}$$

A one-to-one inverse mapping is defined on this set by formulas

$$\forall r \in J^* \quad \kappa^{-1} : \ r \longrightarrow s, \quad s \in J_K, \qquad J_K = \kappa^{-1} J^*.$$

Consider the new grid

$$\widehat{X} : \quad a = \widehat{x}_0 < \widehat{x}_1 < \ldots < \widehat{x}_K = b, \tag{3}$$

where

$$\widehat{x}_i = \xi_{\kappa(i)}, \qquad i \in J_K. \tag{4}$$

Further we consider *virtual* nodes ξ_{-1} and $\widehat{x}_{-1}$ with the properties $\xi_{-1} = \widehat{x}_{-1} < a$. They are virtual in the sense that they serve for the convenience of records, but in the final result they have no effect. By definition we put $X = \|a, b\| \cup \{\xi_{-1}\}$, $\widehat{X} = \{\widehat{x}_i\}_{i \in \{-1, 0, \ldots, K\}}$.

We introduce the functions $\widehat{\omega}_j(t)$, $j \in J'_{K-1}$, $t \in \|a, b\|$, according to the formulas

$$\widehat{\omega}_i(t) = (t - \xi_{\kappa(i)})(\xi_{\kappa(i+1)} - \xi_{\kappa(i)})^{-1}$$

$$\text{for} \quad t \in \|\xi^{\,+}_{\kappa(i)}, \xi_{\kappa(i+1)}\|, \ i \in J_{K-1},$$

$$\widehat{\omega}_i(t) = (\xi_{\kappa(i+2)} - t)(\xi_{\kappa(i+2)} - \xi_{\kappa(i+1)})^{-1}$$

$$\text{for} \quad t \in \|\xi_{\kappa(i+1)}, \xi^{\,-}_{\kappa(i+2)}\|, \ i \in J'_{K-2};$$

$$\widehat{\omega}_i(t) = 0 \quad \text{for} \quad t \in \|a, b\| \| \setminus \|\xi^{\,+}_{\kappa(i)}, \xi^{\,-}_{\kappa(i+2)}\|.$$

It is evident that

$$\widehat{\omega}_i(\xi_{\kappa(i+1)}) = 1 \quad \forall i \in J'_{K-1}.$$

Consider the numbers $\mathbf{p}_{r,s}$, $r \in J'_{K-1}$, $s \in J'_{M-1}$, given by formulas

$$\mathbf{p}_{-1,j} = (\xi_{\kappa(1)} - \xi_{\kappa(0)})^{-1}(\xi_{\kappa(1)} - \xi_{j+1})$$

$$\forall j \in \{\kappa(0) - 1, \kappa(0), \ldots, \kappa(1) - 2\}, \tag{5}$$

$$\mathbf{p}_{i,j} = (\xi_{\kappa(i+1)} - \xi_{\kappa(i)})^{-1}(\xi_{j+1} - \xi_{\kappa(i)})$$

$$\forall j \in \{\kappa(i), \kappa(i) + 1, \ldots, \kappa(i+1) - 1\} \quad \forall i \in J_{K-2}, \tag{6}$$

$$\mathbf{p}_{i,j} = (\xi_{\kappa(i+2)} - \xi_{\kappa(i+1)})^{-1}(\xi_{\kappa(i+2)} - \xi_{j+1})$$

$$\forall j \in \{\kappa(i+1), \kappa(i)+1, \ldots, \kappa(i+2)-2\} \quad \forall i \in J_{K-2}, \tag{7}$$

$$\mathbf{p}_{K-1,j} = (\xi_{\kappa(K)} - \xi_{\kappa(K-1)})^{-1}(\xi_{j+1} - \xi_{\kappa(K-1)})$$

$$\forall j \in \{\kappa(K-1), \kappa(K-1) + 1, \ldots, \kappa(K) - 1\}. \tag{8}$$

The numbers $\mathbf{p}_{r,s}$, $r \in J'_{K-1}$, $s \in J'_{M-1}$, which do not appear in these formulas, are zero.

In the following, we consider a matrix $\mathbf{P}$ of the size $K + 1 \times M + 1$ composed of the numbers $\mathbf{p}_{r,s}$,

$$\mathbf{P} = (\mathbf{p}_{r,s})_{r \in J'_{K-1}, \, s \in J'_{M-1}}. \tag{9}$$

4 Discrete spline-wavelet decomposition

Denote $\mathbf{S}(\widehat{X})$ linear space, which is the linear span of the functions $\widehat{\omega}_j$,

$$\mathbf{S}(\widehat{X}) = \mathcal{L}\{\widehat{\omega}_i(t) \mid \forall t \in \|a,b\| \quad \forall i \in {J'}_{K-1}\}.$$

The space $\mathbf{S}(\widehat{X})$ is called *the discrete space of the first degree splines* (on the grid $\widehat{X}$).

So far $\mathbf{S}(\widehat{X}) \subset C\|a,b\|$, then we can consider the operator P for projection of the space $C\|a,b\|$ on the subspace $\mathbf{S}(\widehat{X})$,

$$Pu = \sum_{i \in {J'}_{K-1}} u(\widehat{x}_{i+1})\, \widehat{\omega}_i \quad \forall u \in C\|a,b\|. \tag{10}$$

Let $Q = I - P$, where I is the identical operator in $C\|a,b\|$.

So, in accordance with (6.1) we obtain the direct decomposition

$$C\|a,b\| = \mathbf{S}(\widehat{X}) + \mathbf{W}, \tag{11}$$

where $\mathbf{W} = \mathbf{QC}\|\mathbf{a},\mathbf{b}\|$.

The space $\mathbf{S}(\widehat{X})$ is called *a basic space* and $\mathbf{W}$ is named *a wavelet space* in decomposition (11).

Let $u \in C\|a,b\|$. Using relation (11), we obtain two representations of the element u

$$u = \sum_{s \in {J'}_{M-1}} c_s \omega_s, \tag{12}$$

and

$$u = \widehat{u} + w. \tag{13}$$

Here

$$\widehat{u} = \sum_{i \in {J'}_{K-1}} a_i \widehat{\omega}_i, \qquad w = \sum_{j \in {J'}_{M-1}} b_j \omega_j,$$

$$a_i = \langle \widehat{g}^{(i)}, u\rangle \quad \forall i \in {J'}_{K-1}, \quad b_j,\ c_s \in \mathcal{R}^1 \quad \forall j,s \in {J'}_{M-1}.$$

Obviously, the relation (6.4) represents *the wavelet decomposition* of the element $u \in C\|a,b\|$, where $\widehat{u} \in \mathbf{S}(\widehat{\mathbf{X}})$, and $w \in \mathbf{W}$.

Due to the linear independence of the system $\{\omega_j\}_{j \in {J'}_{M-1}}$ we get *reconstruction formulas*

$$c_j = \sum_{i \in {J'}_{K-1}} a_i \mathbf{p}_{i,j} + b_j \qquad \forall j \in {J'}_{M-1}. \tag{14}$$

The decomposition formulas have the form

$$b_j = c_j - \sum_{i \in {J'}_{K-1}} \sum_{s \in {J'}_{M-1}} c_s \mathbf{q}_{i,s} \mathbf{p}_{i,j} \qquad \forall j \in {J'}_{M-1}. \tag{15}$$

$$a_i = \sum_{s \in {J'}_{M-1}} c_s \mathbf{q}_{i,s} \qquad \forall i \in {J'}_{K-1}, \tag{16}$$

where

$$\mathbf{q}_{s,j} = \delta_{\kappa(s+1)-1,\, j} \qquad \text{for} \quad s \in {J'}_{K-1},\ j \in {J'}_{M-1}. \tag{17}$$

Consider a *prolongation matrix* $\mathbf{Q}$ of the size $K+1 \times M+1$, composed of elements $\mathbf{q_{s,j}}$,

$$\mathbf{Q} = (\mathbf{q}_{s,j})_{s \in {J'}_{K-1},\, j \in {J'}_{M-1}}.$$

The matrix $\mathbf{Q}$ is the left inverse to the result of transposing for the matrix $\mathbf{P}$, so that

$$\mathbf{QP}^T = I. \tag{18}$$

Here I is the unit matrix of the size $K+1 \times K+1$.

5 Simplification of decomposition and reconstruction formulas

Consider three vectors $\mathbf{a} = (b_{-1}, \ldots, b_{K-1})$, $\mathbf{b} = (b_{-1}, \ldots, b_{M-1})$, $\mathbf{c} = (c_{-1}, \ldots, c_{M-1})$. We will call them *the main, wavelet and source numerical flows*, and the linear spaces of these flows are denoted by $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ respectively.

Ratios

$$\widetilde{u} = \sum_{i \in {J'}_{K-1}} a_i \widehat{\omega}_i, \quad w = \sum_{i \in {J'}_{N-1}} b_i \omega_i, \quad u = \sum_{i \in {J'}_{N-1}} c_i \omega_i$$

are established linear isomorphisms between just introduced spaces and spaces $\mathbf{S}(\widehat{X})$, $\mathbf{W}$, $C\|a,b\|$ so that

$$\mathcal{A} \sim \mathbf{S}(\widehat{X}), \qquad \mathcal{B} \sim \mathbf{W}, \qquad \mathcal{C} \sim C\|a,b\|. \tag{19}$$

In view of the isomorphisms mentioned above, we obtain the direct sum $\mathcal{C} = \mathcal{A} + \mathcal{B}$.

Introduced vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ and matrices $\mathbf{P}$ and $\mathbf{Q}$ allow us to rewrite formulas (14) – (16) in the form

$$\mathbf{c} = \mathbf{b} + \mathbf{P^T a}, \tag{20}$$

$$\mathbf{a} = \mathbf{Qc}, \qquad \mathbf{b} = \mathbf{c} - \mathbf{P^T Qc}.$$

It is clear to see that the elements $[\mathbf{P^T Q}]_{\mathbf{i,j}}$, $i,j \in {J'}_{M-1}$, of matrix product $\mathbf{P^T Q}$ are determined by the formulas

$$[\mathbf{P}^T\mathbf{Q}]_{i,j} = 0 \qquad \text{for } i \in {J'}_{M-1},\ j+1 \in J_M \backslash J^*, \tag{21}$$

$$[\mathbf{P}^T\mathbf{Q}]_{i,j} = \mathbf{p}_{\kappa^{-1}(j+1)-1,\, i} \qquad \text{for } i \in {J'}_{M-1},\ j+1 \in J^*. \tag{22}$$

Thus if $i+1,\ j+1 \in J^*$, then

$$[\mathbf{P}^T\mathbf{Q}]_{i,j} = \delta_{i,j}.$$

Theorem 1. *For the decomposition formulas the next ratios hold*

$$a_i = c_{\kappa(i+1)-1} \qquad \forall i \in J'_{K-1}, \tag{23}$$

$$b_q = 0 \qquad \forall q+1 \in J^*, \tag{24}$$

When $q+1 \in J_M \backslash J^*$, the following equalities are true

$$b_q = c_q - (\widehat{x}_{s+1} - \widehat{x}_s)^{-1}\Big[(\widehat{x}_{s+1} - \xi_{q+1})c_{\kappa(s)-1} +$$

$$+(\xi_{q+1} - \widehat{x}_s)c_{\kappa(s+1)-1}\Big] \tag{25}$$

for all q such that

$$\widehat{x}_s < \xi_{q+1} < \widehat{x}_{s+1} \qquad \forall \, \|\widehat{x}_s, \widehat{x}_{s+1}\| \subset \|a,b\|. \tag{26}$$

The reconstruction formulas are obtained by inversion (25) – (26) and by using relations (23) – (24). As a result we get the following statement.

Theorem 2. *The next equations hold*

$$c_q = b_q + (\widehat{x}_{s+1} - \widehat{x}_s)^{-1}\Big[(\widehat{x}_{s+1} - \xi_{q+1})a_{s-1} +$$

$$+(\xi_{q+1} - \widehat{x}_s)a_s\Big] \tag{27}$$

for all q for which

$$\widehat{x}_s < \xi_{q+1} < \widehat{x}_{s+1} \qquad \forall \, \|\widehat{x}_s, \widehat{x}_{s+1}\| \subset \|a,b\|. \tag{28}$$

Note that the space $\mathcal{B}$ of wavelet flows has the form

$$\mathcal{B} = \{\mathbf{b} \mid \mathbf{b} = (b_{-1}, b_0, \dots, b_{M-1}) \; \forall b_{j-1} \in \mathcal{R}^1 \; \forall j \in J_M \backslash J^*,$$

$$b_{i-1} = 0 \; \forall i \in J^*\}. \tag{29}$$

Thus, the wavelet space is a linear span of the orthonormal [1] system

$$\mathbf{e}_j = ([\mathbf{e}_j]_{-1}, [\mathbf{e}_j]_0, \dots, [\mathbf{e}_j]_{M-1}),$$

where $[\mathbf{e}_j]_i = \delta_{i,j}$, $i,j \in J'_{M-1}$.

[1] In this case we can discuss Euclidean space $\mathcal{R}^{M+1}$ with a standard scalar product.

6 Illustrative example of wavelet decomposition

We put $M = 10$ and on the grid segment $\|a,b\|$, consider the grid

$$a = \xi_0 < \xi_1 < \ldots < \xi_9 < \xi_{10} = b.$$

As a discrete basis (1) we discuss elements $\{\omega_j(t)\}_{j \in J'_{M-1}}$ of the space $C\|a,b\|$,

$$\omega_j(\xi_s) = \delta_{s,j+1}, \qquad s \in J_{10}.$$

Consider the new grid

$$\widehat{X}: \quad a = \widehat{x}_0 < \widehat{x}_1 < \ldots < \widehat{x}_8 = b,$$

where $\widehat{x}_i = \xi_{\kappa(i)}$, $i \in J_8$,and the mapping $\kappa(i)$ is defined by the relations

$$\kappa(0) = 0, \; \kappa(1) = 1, \; \dots, \kappa(5) = 5,$$

$$\kappa(6) = 8, \; \kappa(7) = 9, \; \kappa(8) = 10.$$

So, the considered case is $K = 8$, and the subset J^* of the set J_{10}, obtained by formula (2), has the form

$$J^* = \{0,1,2,3,4,5,8,9,10\}.$$

It is clear to see$J_{10} \backslash J^* = \{6,7\}$.

We introduce the functions $\widehat{\omega}_j(t)$, $j \in J'_7$, according to formulas (5) – (8) and consider the vectors

$$\widehat{\omega}(t) = (\widehat{\omega}_{-1}(t), \widehat{\omega}_0(t), \dots, \widehat{\omega}_7(t))^T,$$

$$\omega(t) = (\omega_{-1}(t), \omega_0(t), \dots, \omega_9(t))^T.$$

From formulas (5) – (9) it follows that $\widehat{\omega}(t) = \mathbf{P}\omega(\mathbf{t})$ so that the calibration relations are determined by the matrix $\mathbf{P}$.

Therefore

$$\widehat{\omega}_i(t) = \omega_i(t) \quad \text{for } i \in \{-1,0,1,2,3\},$$

$$\widehat{\omega}_4(t) = \omega_4(t) + \frac{\xi_8 - \xi_6}{\xi_8 - \xi_5} \cdot \omega_5(t) + \frac{\xi_7 - \xi_6}{\xi_8 - \xi_5} \cdot \omega_6(t),$$

$$\widehat{\omega}_5(t) = \frac{\xi_6 - \xi_5}{\xi_8 - \xi_5} \cdot \omega_5(t) + \frac{\xi_7 - \xi_5}{\xi_8 - \xi_5} \cdot \omega_6(t) + \omega_7(t),$$

$$\widehat{\omega}_i(t) = \omega_{i+2}(t) \quad \text{for } i \in \{6,7\}.$$

The operation for projection of the space $C\|a,b\|$ of initial flows on the space $\mathbf{S}(\widehat{X})$ is defined by formulas (10), (16) and (23).

Ultimately, this projection means the application of prolongation matrix $\mathbf{Q}$ to the original flow $\mathbf{c} \in \mathcal{C}$: $\mathbf{a} = \mathbf{Q}\mathbf{c}$. According to formulas (17) in this case, the matrix $\mathbf{Q}$ forms the flow $\mathbf{a}$ so that

$$a_i = c_i \qquad \text{for} \quad i \in \{-1,0,1,2,3,4\}, \tag{30}$$

$$a_i = c_{i+2} \qquad \text{for} \quad i \in \{7, 8\}. \tag{31}$$

It is easy to see that $\mathbf{QP}^T = I$, where I is the unit square matrix of the size 9×9. By (21) – (22) it follows that square matrix $\mathbf{P}^T\mathbf{Q}$ has the size 11×11. Note that the components of the wavelet $\mathbf{b} = \mathbf{c} - \mathbf{P}^T\mathbf{Qc}$ is obviously zero if their numbers j are such that the numbers $j+1$ are contained in the set J^*. Wavelet flow $\mathbf{b}$ is obtained from the source flow $\mathbf{c}$ using formulas (25). In this case, we obtain

$$b_j = 0 \qquad \text{for} \quad j \in \{-1, 0, 1, 2, 3, 4, 7, 8, 9\}, \tag{32}$$

$$b_5 = -\frac{\xi_8 - \xi_6}{\xi_8 - \xi_5} c_4 + c_5 - \frac{\xi_6 - \xi_5}{\xi_8 - \xi_5} c_7, \tag{33}$$

$$b_6 = -\frac{\xi_8 - \xi_7}{\xi_8 - \xi_5} c_4 + c_6 - \frac{\xi_7 - \xi_5}{\xi_8 - \xi_5} c_7. \tag{34}$$

Formulas (30) – (34) are decomposition formulas.

Using the formula (20), we obtain formulas for the reconstruction of the original flow $\mathbf{c}$:

$$c_j = a_j + b_j \qquad \text{for} \quad j \in \{-1, 0, 1, 2, 3, 4\},$$

$$c_5 = \frac{\xi_8 - \xi_6}{\xi_8 - \xi_5} \cdot a_4 + \frac{\xi_6 - \xi_5}{\xi_8 - \xi_5} \cdot a_5 + b_5,$$

$$c_6 = \frac{\xi_8 - \xi_7}{\xi_8 - \xi_5} \cdot a_4 + \frac{\xi_7 - \xi_5}{\xi_8 - \xi_5} \cdot a_5 + b_6,$$

$$c_j = a_{j-2} + b_j \qquad \text{for} \quad j \in \{7, 8, 9\}.$$

7 Further supposition

The calculation of the wavelet decomposition involves two stages: the first stage is the implementation of decomposition formulas, and the second stage is the implementation of the reconstruction formulas.

The implementation of decomposition formulas contains two tasks: looking for the main flow and looking for the wavelet flow. The first of these tasks is usually more important than the second one because in most cases, it is the main flow that gives an idea of the nature of the source flow.

It is further believed that the considered computational system (CS) is a discrete action computing system which works synchronously (in cycles). Any considered time interval has integer length (see [5]).

When the above algorithms are implemented, you have to extract the element from some array (i.e. you assign its values to some intermediate variable), immerse the element to another array (i.e. assign the value of the intermediate variable to element array), and also to do certain arithmetic and logical operations.

We introduce some notation. Let $\mathbf{A}$ be an array. The elements A_i of the array are numbers of type $f_{\mathbf{A}}$. For simplicity, we discuss the numbers of one type. For example, we can assume that $f_{\mathbf{A}} = f$, where f means the type of `real`.We will not consider the specific representation of the types $f_{\mathbf{A}}$ and f in CS.

Let's assume that $\breve{T}_{\mathbf{A}}$ is the time (i.e. the length of the time segment[2]) extracting the i-th element A_i of array $\mathbf{A}$ into a simple (auxiliary) variable; the time required to immerse a simple variable value in the element A_i of this array we denote by $\widehat{T}_{\mathbf{A}}$[3].

For brevity, hereafter, the storage for flows $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ is denoted by the same symbols. Therefore $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ are also storage (arrays) for the mentioned flows. Such agreement applies to other similar objects: X, $\widehat{X}$, J_s, J^*, etc. All arrays considered further are discussed as dynamically extensible (i.e. the length of the array in advance is not fixed, when we add an element to an array, its length increases per unit).

8 Wavelet decomposition (single processor approach)

Consider the decomposition formulas (23) – (24) and (25) – (26). You can imagine a number of calculation options for these kinds of formulas. Consider one of these, assuming that

$$a = 0, \quad b = M, \qquad \xi_i = i, \tag{35}$$

Thus

$$X = \{-1, a = 0, 1, 2, \ldots, M-1, M = b\}.$$

In this case $\widehat{x}_i = \kappa(i)$, $i \in \{0, 1, \ldots, K\}$, and

$$\widehat{X} = \{-1, a = \kappa(0), \kappa(1), \kappa(2), \ldots, \kappa(K-1), \kappa(K) = b\}.$$

It is evident that $X = J_M$ $\widehat{X} = J^*$.

Using (35) in formulas (25) – (26), we obtain

$$b_q = c_q - (\kappa(i+1) - \kappa(i))^{-1} \Big[(\kappa(i+1) - q - 1) c_{\kappa(i)-1} +$$
$$+ ((q + 1 - \kappa(i)) c_{\kappa(i+1)-1} \Big], \tag{36}$$

where

$$\kappa(i) + 1 \le q + 1 \le \kappa(i+1) - 1 \quad \forall \|\widehat{x}_i, \widehat{x}_{i+1}\| \subset \|a, b\|. \tag{37}$$

[2]Recall that considered time is discrete, the unit of time is equal to the length of the clock cycle

[3]We believe that the mentioned times do not depend on the number i of the element in question, but may depend on the type of array elements and on its length

Let the implementation of the algorithm for finding $j = \kappa(i)$ require τ_i units of time; we assume that the additive operation requires t_a time units, and multiplicative t_m units of time.

For clarity, as a rule, we indicate the names of intermediate variables, although their presence implied (for example, instead of assigning $j := \kappa(i+1)$ with further use of the simple variable j we will talk "calculate $\kappa(i+1)$").

The implementation of the decomposition will be represented as sequence stages. Consider the $i+1$-th stage of the decomposition process.

To indicate the state of the arrays at the i-th stage, we will use i as a subscript.

Before the beginning of the $i+1$-th stage, the state of the arrays is characterized as follows:

a/ calculated the value of $\kappa(i)$ (and saved in some simple variable)

b/ array X_i is represented as

$$X_i = \{0, 1, 2, \ldots, i-1, i\},$$

and array $J_i^* = \widehat{X}_i$ looks like

$$\widehat{X}_i = \{a = \kappa(0), \kappa(1), \kappa(2), \ldots, \kappa(i)\},$$

c/ array $\mathbf{a}$ is filled up to the element $a_{\kappa(i-1)}$, so that

$$\mathbf{a}_i = \{a_{\kappa(0)}, a_{\kappa(1)}, \ldots, a_{\kappa(i-1)}\},$$

d/ array $\mathbf{b}$ is filled up to the element $b_{\kappa(i-1)-1}$, those.

$$\mathbf{b}_i = \{b_{\kappa(i_0+1)}, \ldots, b_{\kappa(i_1)-1}\},$$

where $i_0 = \min\{i \mid i \in X \backslash \widehat{X}\}$, $i_1 = \max\{i \mid i \in X \backslash \widehat{X}\}$,

e/ calculated the value of $c_{\kappa(i)-1}$.

Carrying out the $i+1$ th stage consists of the following actions.

1. First, we compute $\kappa(i+1)$; it will require $\tau_i + t_a$ units of time.

2. Connect to the array $\widehat{X}_i$ the next element $\kappa(i+1)$; this will require $\widehat{T}_{\widehat{X}}$ units of time, and the array will take the form of the algorithm.

3. Calculation and extraction of the element $c_{\kappa(i+1)-1}$ from array $\mathbf{c}$ additionally requires $t_a + \breve{T}_{\mathbf{c}}$ units of time.

4. Adding the element $a_i = c_{\kappa(i+1)-1}$ to the array $\mathbf{a}$ will require $\widehat{T}_{\mathbf{a}}$ units of time (recall that the value of $\kappa(i+1)-1$ already computed and placed in a simple variable whose name is not mentioned in accordance with the agreement).

5. Using formulas (36) – (37), we calculate b_q for each

$$q \in \{\kappa(i), \kappa(i)+1, \ldots, \kappa(i+1)-2\} \qquad (38)$$

(notice, that the set of indices (38) is not empty, because $q+1 \in J_M \backslash J^*$).

Since the elements of $\kappa(i)$, $\kappa(i+1)$ and $c_{\kappa(i)-1}$ are already calculated, then you only need to extract the element $c_{\kappa(i+1)-1}$ from array $\mathbf{c}$ and make up the difference $\kappa(i+1) - \kappa(i)$; this will require $\breve{T}_{\mathbf{c}} + t_a$ units of time.

To calculate b_q, we set $q = \kappa(i) + j$, so that

$$b_{\kappa(i)+j} = c_{\kappa(i)+j} -$$
$$-(\kappa(i+1)-\kappa(i))^{-1}\Big[(\kappa(i+1)-\kappa(i)+j-1)c_{\kappa(i)-1} +$$
$$+(j+1)c_{\kappa(i+1)-1}\Big], \qquad (39)$$

and create a loop on $j \in \{0, 1, \ldots, \kappa(i+1)-\kappa(i)-2\}$.

The j-th iteration of this cycle will require certain time units.

1). Extraction of $c_{\kappa(i)+j}$ from the array $\mathbf{c}$ requires $t_a + \breve{T}_{\mathbf{c}}$ time units. 2). The expression in square brackets of (39) require four additive operations and two multiplicative (previously performed operations naturally in this counting is not counted). Thus they require $4t_a + 2t_m$ units of time.

3). Outside, the square brackets will need to perform one multiplicative and one additive operation. They will take $t_a + t_m$ time units.

4). The resulting value $b_q = b_{\kappa(i)+j}$ is needed to immerse in the array $\mathbf{b}$. That will require $\widehat{T}_{\mathbf{b}}$ units of time.

So, for the implementation of a single iteration of the loop in j required $6t_a + 3t_m + \breve{T}_{\mathbf{c}}$ units of time, and such iterations total $\kappa(i+1) - \kappa(i) - 1$ (obviously, there are no iterations in the case when $\kappa(i+1) = \kappa(i)+1$). Taking into account the preparatory operations mentioned above to find all required values of b_q at the $i+1$-th stage will be required $\breve{T}_{\mathbf{c}} + t_a + (6t_a + 3t_m + \breve{T}_{\mathbf{c}} + \widehat{T}_{\mathbf{b}})(\kappa(i+1) - \kappa(i) - 1)$ units of time.

Now it is clear that for the implementation of the $i+1$-th stage as a whole required

$$\tau_i + 3t_a + \widehat{T}_{\widehat{X}} + \breve{T}_{\mathbf{c}} + \widehat{T}_{\mathbf{a}} +$$
$$+(6t_a + 3t_m + \breve{T}_{\mathbf{c}} + \widehat{T}_{\mathbf{b}})(\kappa(i+1) - \kappa(i) - 1) \quad (40)$$

units of time. Summing (40) for $i \in \{0, 1, \ldots, K-1\}$, we obtain the next assertion.

Theorem 3. *Time T required for the realization algorithm decomposition with the OPS is calculated by the formula*

$$T = \sum_{i=0}^{K-1} \tau_i + K(3t_a + \widehat{T}_{\widehat{X}} + 2\breve{T}_{\mathbf{c}} +$$
$$+\widehat{T}_{\mathbf{a}}) + (6t_a + 3t_m + \breve{T}_{\mathbf{c}} + \widehat{T}_{\mathbf{b}})(M - K).$$

9 Wavelet reconstruction (single processor approach)

The evaluation of the reconstruction time is carried out similarly. Provided (35) formulas (27) – (28) take the form (as before, the division on $\kappa(i+1)-\kappa(i)$ is distributed between the terms to reduce rounding errors)

$$c_j = \frac{\kappa(i+1)-j-1}{\kappa(i+1)-\kappa(i)}\cdot a_{i-1}+\frac{j+1-\kappa(i)}{\kappa(i+1)-\kappa(i)}\cdot a_i+b_j \tag{41}$$

$$\forall j \in J'_{M-1},$$

where i is defined by relation (28),

$$\kappa(i) \le j+1 \le \kappa(i+1). \tag{42}$$

The search for the numbers c_j is done in a loop by $j \in \{-1, 0, 1, \ldots, M-1\}$.

Let the time required to assign a value for simple variable is equal to t_s. Let the comparison operation requires t_c units of time.

Theorem 4. *Quantity T^* of time required for realization reconstruction algorithm with the OPS, is calculated by the formula*

$$T^* = (8t_a + 4t_c + 4t_m + 2\breve{T}_{\mathbf{a}} + \breve{T}_{\mathbf{b}} + \widehat{T}_{\mathbf{c}})\times$$

$$\times(M+1) + \sum_{i=0}^{K} \tau_i + 2t_s. \tag{43}$$

Proof. We divide the proof into five steps.Each of which deals with a specific part in the implementation of the proposed algorithm.

0). Before the start of the cycle we assume $i = 0$, $j = -1$ and find $\kappa(1)$. This will require $2t_s + \tau_1$ units of time.

1). First we calculate $\kappa(i+1)$. It will require $t_a + \tau_{i+1}$ units of time.

2). In each iteration of the cycle, we check equality $j+1 = \kappa(i+1)$, and if it is true, then we set $i = i+1$. This will require $t_a + t_c$ units of time. The time τ_{i+1} of calculation of $\kappa(i+1)$ will be accounted later.

3). Now we calculate c_j using the formula (41). We need to make four multiplicative and five additive operations, two extracts from the array $\mathbf{a}$, one extract from array $\mathbf{b}$, and immersing the result in the array $\mathbf{c}$. So we need to use $4t_m + 5t_a + 2\breve{T}_{\mathbf{a}} + \breve{T}_{\mathbf{b}} + \widehat{T}_{\mathbf{c}}$ time units.

4). We end the iteration of the loop, setting $j = j+1$. Here you need $t_a + t_c$ units of time.

5). We check the equality $j+1 = M$. If it is true, then the cycle is over, the initial flow is found. If it is not correct, we need to go to the beginning of the cycle. This will require $t_c + t_l$ units of time.

So, one iteration of the loop will be used $8t_a + 4t_c + 4t_m + 2\breve{T}_{\mathbf{a}} + \breve{T}_{\mathbf{b}} + \breve{T}_{\mathbf{c}}$ (excluding sometimes appearing add τ_{i+1}). The number of loop iterations is the number of elements in the stream $\mathbf{c}$, i.e. equals $M+1$. Considering previously added calculations of the values of τ_{i+1}, we see what's on the implementation of the loop (along with preparatory assignments zero stage) will require units of time. Thus we obtain equality (43). This concludes the proof.

10 On wavelet interpretation with parallel system

In the following, parallelization of decomposition and reconstruction algorithms is considered. We discuss the situation of the unlimited concurrency. Therefore 1) there are so many parallel processors (cores), as needed for the considered tasks, 2) there is potentially unlimited memory (i.e. again, there is as much memory as it is needed for tasks), 3) processors have free access to the memory without collisions.

Let $\breve{T}_{\mathbf{a}}$, $\breve{T}_{\mathbf{b}}$, $\breve{T}_{\mathbf{c}}$ be the number of time units required to retrieve an element from arrays $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ respectively.[4]

Let $\widehat{T}_{\mathbf{a}}$, $\widehat{T}_{\mathbf{b}}$, $\widehat{T}_{\mathbf{c}}$ be the number of time units, which are necessary for immersing an element into arrays $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ respectively.[5]

Let t_a, t_m be the number of time units for (parallel) additive and multiplicative operations.

Let t_s, t_c, t_l, t_* be the number of time units for (parallel) assignment, comparison, logical transition and exchange between "neighboring" processors, respectively.

Finally, assume that the mapping calculation $\kappa(i)$ takes τ time units.

Note that in parallel computing the group of simultaneous operations are called *deck*. The number of these operations is called the *width of the deck*.In our case, it is assumed that the implementation time of the deck may be different. This allows you to use a deck of complex structure. Such decks are called *a stratum*. They may be considered as small parallel forms.

The time of implementation (runtime) of the stratum is called *the height of the stratum*. In parallel form, the stratums are ordered so that the operands for

[4]As before considered time is discrete, the unit of time is equal to the length of the clock cycle.

[5]Recall once more that the mentioned times do not depend on the number i of the element in question, but may depend on the type of array elements and on its length

the operations of each stratum are obtained on previous stratums, or are the original algorithm data. Parallel algorithm (parallel form) is the ordered system of stratums. The sum of the stratum heights for the parallel algorithm is called *the height of the parallel form*. In conditions unlimited concurrency we will try to find the parallel form of small height.

11 Calculating the wavelet decomposition (parallel processor approach)

The implementation of formulas (36) – (37) on PMS can be discussed in different ways. Consider the following calculation steps.

1. Using the stratum of height τ and width $K+1$, we find all the values of $\kappa(i)$, $i \in J'_{K-1}$. Thus we define the subset J^* of the set J'_{M-1}.

2. We perform two additive operations (we find $j+1$ and $i+1$). Then we exchange two parts of information (to receive $\kappa(j+1)$ and $c_{\kappa(i+1)-1}$). Now we extract numbers $c_{\kappa(i)-1}$ and immerse them in an array $\mathbf{a}$. At last we do two additive operations. For all of these operations we have to construct a stratum with height $2t_*+\breve{T}_\kappa+\breve{T}_\mathbf{c}+\widehat{T}_\mathbf{a}+4t_a$ and width $M+1$.

3. Finding b_q by formulas (24), (36) – (37) will require extraction of elements from array $\mathbf{c}$ and κ, immersing results into the array $\mathbf{c}$, as well as five additive and four multiplicative operations. Thus, the height of this stratum is $\breve{T}_\mathbf{c}+\breve{T}_\kappa+\widehat{T}_\mathbf{b}+5t_a+4t_m$, and its width is $M+1$.

Thus, the following statement is established.

Theorem 5. *For the realization of decomposition formulas there is a parallel form of height*

$$H=\tau+2t_*+2\breve{T}_\kappa+2\breve{T}_\mathbf{c}+\widehat{T}_\mathbf{a}+\widehat{T}_\mathbf{b}+9t_a+4t_m \quad (44)$$

with a width of $W=M+1$.

12 Calculating of the wavelet reconstruction (parallel processor approach)

Consider a parallel implementation of formulas (41) – (42). We know that with successive calculations, finding the numbers c_j was carried out in a loop on $j \in J'_{M-1}$. We notice, that loop iterations can be considered independently from each other. Therefore they can be calculated by a parallel system effectively. Here we will need

1. Extraction of elements from arrays $\kappa(i)$, $\mathbf{a}$, $\mathbf{b}$,what is required the height of stratum $\breve{T}_\kappa+\breve{T}_\mathbf{a}+\breve{T}_\mathbf{b}$ and widths $M+1$.

2. Six additive operations and four multiplicative operations, and besides, two exchanges between neighboring processors for sending values a_{i-1} and $\kappa(i+1)$. This can be done by the parallel form of height $6t_a+4t_m+2t_*$.

3. Immersing the result in the array $\mathbf{c}$ requires a parallel form of height $\widehat{T}_\mathbf{c}$ and width $M+1$.

Thus we have proved the next assertion.

Theorem 6. *For realization of reconstruction algorithm with PMS there is a parallel form of height*

$$H^*=\breve{T}_\kappa+\breve{T}_\mathbf{a}+\breve{T}_\mathbf{b}+6t_a+4t_m+2t_*+\widehat{T}_\mathbf{c}$$

and width $W^*=M+1$.

13 Conclusion

The numerical implementation of the wavelet decomposition requires the original grid X and the nested grid $\widehat{X}$. This construction is usually determined by the original flow $\mathbf{c}$. It may require significant computer resources. Notice that the algorithm for the construction of the embedded grid used here is optimal in a certain sense. The disadvantage of this algorithm is that it is an essentially sequential process. Though in real situations it can be handled as OPS, and on PMS.

Taking into account the obtained algorithms we consider their implementation with OPS and PMS. In the situation of the unlimited concurrency the duration of calculation with PMS does not depend on the data volume (i.e. it does not depend on the length of the initial flow), on the other hand, the duration of calculation with OPS is directly proportional to the data volume.

The application of a sequential algorithm (or parallel algorithm of small width) is preferable in the case of the arrival of the initial flow in real time. In this case, the entire flow is not yet received. If the source flow has already been received, then its processing on PMS can be very effective due to the splitting flow into sufficiently long fragments, the number of which is equal to the number ofparallel processors. The processing each of these parts carried out sequentially assigned to this part computational module with the subsequent connection of these parts in the original order. These questions will be discussed in the wider version of this paper.

Acknowledgements: This work was partly supported by RFBR Grant 15-01-08847.

References:

[1] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, 1999.

[2] Novikov I.Ya., Protasov V.Yu., Skopina M.A. *Wavelet Theory,* AMS, Translations Mathematical Monographs, V. 239, 2011.

[3] Yu. K. Demyanovich, Wavelet decompositions on nonuniform grids, *Am. Math. Soc. Transl. Ser.* 2, 222, 2008, pp. 2342.

[4] Yu. K. Demyanovich, Wavelets on a manifold, Dokl. Math. 79, No. 1, 2009, pp. 2124.

[5] Yu. K. Demyanovich and A.Yu.Ponomareva, Adaptive Spline-Wavelet Processing of a Discrete Flow, J. Math. Science, New York 210, No. 4, 2015, pp.371-390.

[6] Yu.K.Dem’yanovich, A.S.Ponomarev, On realization of the Spline-Wavelet Decomposition of the First Order, *J. of Math. Science,* 224, No.6, 2017, pp.833-860.

[7] R. Courant, ”Variational methods for the solution of problems of equilibrium and vibrations, *Bulletin of the American Mathematical Society,* 49, 1943, pp.123.

[8] S.G. Michlin, *Approximation auf dem Kubischen Gitter*, Berlin, 1970.

[9] S.G. Michlin, *Konstanten in einige Ungleichungen der Analysis*, Leipzig, 1981.

[10] J.N. Reddy, *An Introduction to the Finite Element Method*, McGraw-Hill, 2006.

[11] O.C. Zienkiewicz, R.L. Taylor, J.Z. Zhu, *The Finite Element Method: Its Basis and Fundamentals,* Butterworth-Heinemann, 2005.

[12] K.J. Bathe, *Finite Element Procedures*, Cambridge, MA: Klaus-Jurgen Bathe, 2006.

[13] Ivo Babuska, Uday Banerjee, John E. Osborn, Generalized Finite Element Methods: Main Ideas, Results, and Perspective, *International Journal of Computational Methods* 1 (1), 2004, pp.67-103.

[14] G. R. Liu, K. Y. Dai, T. T. Nguyen, A smoothed finite element method for mechanics problems, *Comput. Mech.* 39, 2007, pp.859 - 877.

[15] G. R. Liu, T. T. Nguyen, K. Y. Dai and K. Y. Lam, Theoretical aspects of the smoothed finite element method (SFEM), *Int. J. Numer. Meth. Engng.* 71, 2007, pp.902-930.

[16] G.R. Liu, *Smoothed Finite Element Methods*, CRC Press, 2010.

[17] T. Nguyen-Thoi, G. R. Liu and H. Nguyen-Xuan, An n-sided polygonal edge-based smoothed finite element method (nES-FEM) for solid mechanics, *Int. J. Numer. Meth. Biomed. Engng.* 2010, (www.interscience.wiley.com). DOI: 10.1002/cnm.1375.

[18] Vahid Shobeiri, Structural Topology Optimization Based on the Smoothed Finite Element Method, *Latin American Journal of Solids and Structures*, 13, 2016, pp.378-390.

[19] W. Zeng, G.R. Liu, Smoothed finite element methods (S-FEM): An overview and recent developments, *Archives of Computational, Methods in Engineering*, 2016. DOI: 10.1007/s11831-016-9202-3.

[20] G.R. Liu, G.R. Zhang, Edge-based Smoothed Point Interpolation Methods, *International Journal of Computational Methods*, 5(4), 2008, pp.621-646.

[21] Z.Q. Zhang, G.R. Liu, Upper and lower bounds for natural frequencies: A property of the smoothed finite element methods, *International Journal for Numerical Methods in Engineering*, 84, Issue: 2, 2010, pp.149-178.