PROBLEMY MECHATRONIKI
UZBROJENIE, LOTNICTWO, INŻYNIERIA BEZPIECZEŃSTWA

ISSN 2081-5891
E-ISSN 2720-5266

13, 3 (49), 2022, 27-42

PROBLEMS OF MECHATRONICS
ARMAMENT, AVIATION, SAFETY ENGINEERING

The Study of the Possibility of Applying Parallel Programming to the Algorithms of Space-Time Adaptive Processing

Błażej ŚLESICKI[*1], Adam KAWALEC[2], Anna ŚLESICKA[2]

[1]*The Polish Air Force University*
35 Dywizjonu 303 Str., 08-521 Dęblin, Poland
[2]*Military University of Technology*
Faculty of Mechatronics, Armament and Aerospace,
2 Sylwestra Kaliskiego Str., 00-908 Warsaw, Poland
[*]*Corresponding author's e-mail address and ORCID*:
b.slesicki@law.mil.pl; https://orcid.org/0000-0002-0857-1081

Received: January 10, 2022 / Revised: March 12, 2022 / Accepted: July 22, 2022 / Published: September 30, 2022

DOI 10.5604/01.3001.0016.0048

Abstract. The article presents the description, assumptions and subsequent steps of the space-time adaptive processing (STAP) algorithms used as a signal processing tool in radars. The possibilities of object detection using the Sample Matrix Inversion (SMI) and Data Domain Least Squares (DDLS) algorithms were compared and showned. The article shows the impact of the use of parallel programming on the computation time of both algorithms. The main aim of this study was to propose an efficient method for the real-time implementation of the STAP algorithm in airborne radar systems. The idea of using parallel programming in STAP, supported only by the preliminary research results presented above, gives a real chance for the casual implementation of the STAP algorithm in a radar operating in close to real time mode.
Keywords: space-time adaptive processing, radar signal processing, radar

1. INTRODUCTION

Space-time adaptive processing is a crucial technology in the processing of radar signals received by the operating airborne radar in an environment where suppression of clutter and jammer is a challenge. Currently, the search for solutions for the optimal implementation of STAP technology for real-time operation is one from the major branches of STAP research.

Several studies on the implementation of STAP in real-time airborne radar have confirmed the existence of three main problems. The first, essential, is to meet the requirement of access to a huge number of independent and identically distributed range samples. The fulfillment of this condition guarantees an accurate estimation of the interference covariance matrix. Another drawback is the high computational costs associated with inverting the aforementioned interference covariance matrix. Finally, an enormous amount of data generated during the collection and pre-processing of the received signal reflected from the earth's surface should be stored in the processor's memory [1-7].

The possibility of implementing STAP algorithms based on statistical methods of estimating the interference covariance matrix in the MATLAB environment with the use of parallel calculations was the main intention of the authors of this article. So far, attempts have been made to implement STAP algorithms on digital signal processors' platforms. It turned out to be effective, although in fact it was supposed to show the finished product as well as to show the advantages of a digital signal processor (DSP) in real operation [8-11].

Currently, the direction of research on the use of parallel computing in STAP is the use of the CUDA (*Compute Unified Device Architecture*) architecture. CUDA is a computing architecture that enables the use of the computing power of multi-core processors (mainly graphics cards). The article [12] presents the implementation of the EFA STAP (*Extended Factored Algorithm STAP*) algorithm on the CUDA platform with the use of OpenMP. The EFA STAP algorithm is also a STAP algorithm based on the statistical method of estimating the interference covariance matrix. In addition, general implementations that can support variable-sized datasets were presented, and a general study of the tradeoffs between the power and performance of a full STAP pipeline of different architectures was done.

The aim of the article is to propose STAP methods, which use parallelisation of operations with the use of program loops, so as to enable effortless implementation in radar operating in near real time mode.

2. STAP ALGORITHMS

The section presents the steps of processing the received signal reflected from the earth's surface in the STAP technique. It was assumed that the antenna array consists of N isotropic antennas evenly spaced.

The antenna array is transmitting M pulses with the pulse repetition frequency of f_r. First, the SMI STAP algorithm is described. Hence, for the given ranging cell p, it is possible to indicate a single snapshot of the radar cube of the received data:

$$\boldsymbol{X}_p = \begin{bmatrix} x_{p,0,0} & x_{p,1,0} & x_{p,N-1,0} \\ x_{p,0,1} & x_{p,1,1} & x_{p,N-1,1} \\ \vdots & \vdots & \vdots \\ x_{p,0,M-1} & x_{p,1,M-1} & x_{p,N-1,M-1} \end{bmatrix} \quad (1)$$

The following figure shows the radar data cube model.

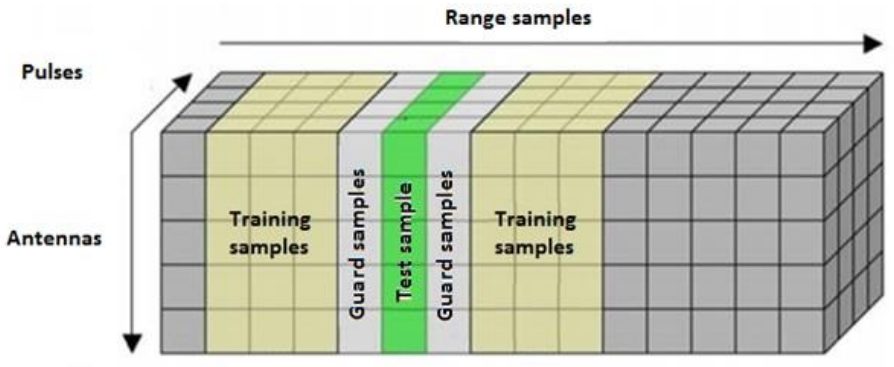

Fig. 1. The radar data cube

The received signal includes reflections from objects illuminated by the main beam, but also by side lobes, hence the main task of the STAP technique is to remove undesired components of the received signal, which are clutter and noise. In the SMI STAP algorithm, the dependence on the optimal filter is given, which suppresses the sum of clutter and noise while amplifying the signal from the object [4]:

$$\boldsymbol{W}_{opt} = \boldsymbol{R}_u^{-1}\boldsymbol{v} \quad (2)$$

where $\boldsymbol{R}_u$ is the estimated interference covariance matrix expressed as $\boldsymbol{R}_u = E[\boldsymbol{X}^T\boldsymbol{X}]$ (T is the Hermitian conjugation of the matrix, E is the expected value), $\boldsymbol{X}$ is the training range samples, and $\boldsymbol{v}$ is the space-time steering vector of the considered object and it is expressed as $\boldsymbol{v} = \boldsymbol{v}_t \otimes \boldsymbol{v}_s$ ($\otimes$ denotes the Kronecker product).

The individual formulas for the temporal steering vector $\boldsymbol{v}_t$ and the spatial steering vector $\boldsymbol{v}_s$ are given as [4]:

$$\boldsymbol{v}_t = \frac{1}{\sqrt{M}} \begin{bmatrix} 1 \\ e^{j2\pi f_d} \\ \vdots \\ e^{j2\pi(M-1)f_d} \end{bmatrix} \tag{3}$$

$$\boldsymbol{v}_s = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 \\ e^{j2\pi f_a} \\ \vdots \\ e^{j2\pi(N-1)f_a} \end{bmatrix} \tag{4}$$

The spatial frequency is denoted as f_a and the Doppler frequency as f_d. The range samples surrounding the range sample, being tested for the presence of the object, are used to estimate the interference covariance matrix $\boldsymbol{R}_u$. The rank of the estimated interference covariance matrix $\boldsymbol{R}_u$ is given as [1]:

$$r_{clut} = N + (M-1)\beta \tag{5}$$

where the parameter β is expressed as $\beta = (2V_a)/(f_r d)$, V_a is the speed of the platform and d is the inter-element spacing of the antenna.

The second algorithm described is DDLS STAP. The use of the same antenna array and the same number of transmitted pulses as for the SMI STAP method was assumed. The first step is to create the T matrix representing the training data necessary for the interference estimation [8-10]:

$$\boldsymbol{T} = \begin{bmatrix} x_{0,0,0} & \cdots & x_{0,N-1,M-1} \\ \vdots & \ddots & \vdots \\ x_{p,0,0} & \cdots & x_{p,N-1,M-1} \end{bmatrix} \tag{6}$$

where P is the number of range samples, N is the number of antennas and M is the number of pulses. The purpose of the DDLS method, like the SMI method, is to find the vector $\boldsymbol{W}$ such that:

$$[\boldsymbol{T}][\boldsymbol{W}] = 0 \tag{7}$$

However, to keep the constant gain in the direction from which the signal comes from the object, a row representing the hypothetical object position should be added to the matrix $\boldsymbol{T}$:

$$\boldsymbol{H} = [\boldsymbol{v}_t \otimes \boldsymbol{v}_s; \quad \boldsymbol{T}] \tag{8}$$

As a result:

$$[\boldsymbol{H}][\boldsymbol{W}] = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix} \tag{9}$$

or

$$[\boldsymbol{H}][\boldsymbol{W}] = \boldsymbol{B} \tag{10}$$

The system of equations (10) written above is undefined, i.e., it has more unknowns than equations. Hence, solutions are sought using optimisation methods. In connection with the above, the optimisation problem is given as [8-10]

$$min(\|\boldsymbol{W}\|_{\mathbf{2}})^{\mathbf{2}} \tag{11}$$

so as to

$$[\boldsymbol{B}] = [\boldsymbol{H}][\boldsymbol{W}] \tag{12}$$

In [6], the above optimisation problem leads to the solution of a given formula:

$$\boldsymbol{W} = \boldsymbol{H}^{\mathrm{T}}\left(\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}}\right)^{-1}\boldsymbol{B} \tag{13}$$

The table below compares the properties of both SMI and DDLS methods.

Table 1. Comparison of SMI and DDLS methods

Parameter	**SMI**	**DDLS**
The dimension of the matrix being inverted	Inversion of the matrix $\boldsymbol{R}_{\mathbf{u}}$ with NM x NM dimension	Inversion of the matrix $\boldsymbol{H}$ with $(P+1)$ x $(P+1)$ dimension
Number of the required training samples P	$P > 2(N + (M\text{-}1)\beta)$	
Required amount of memory	The matrix $\boldsymbol{R}_{\mathbf{u}}$ should be stored with NM x NM dimension	The matrix $\boldsymbol{H}$ should be stored with $(P+1)$ x $(P+1)$ dimension
Computation costs	$(NM)^3 + M(NM)^2$ operation	$P^3 + 2NM(P)^2$ operations for a single hypothetical object

3. APPLICATION OF PARALLEL PROGRAMMING

Parallel programming is a commonly used form of program execution. The essence of parallel programming is to execute several commands, often called threads, at the same time.

The main purpose of using parallel programming is to increase the computational efficiency and to shorten the time needed to solve a specific single computational task. In the article, a special toolbox built into the MATLAB environment – Parallel Computing Toolbox (PCT) was used for research on the use of parallel programming. The figure below shows an illustration of the division of a computational task performed in a loop into 3 threads using PCT [11-13].

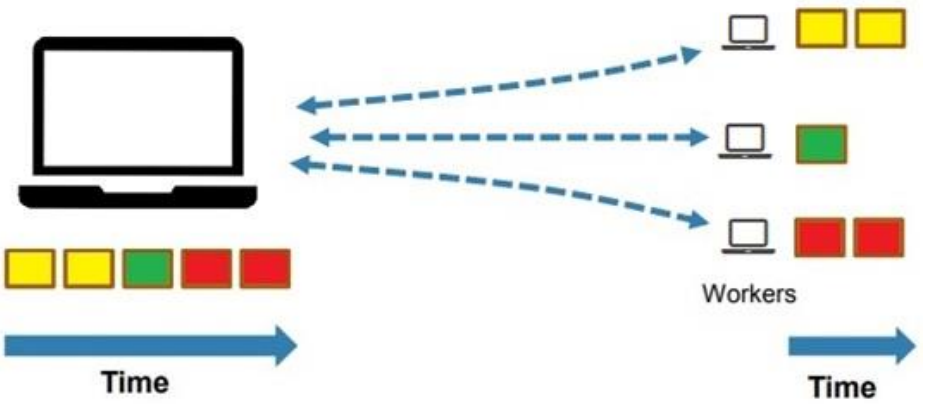

Fig. 2. The division of a computational task performed in a loop into four 3 threads using PCT [11-13]

The article proposes the use of parallel programming in both above-mentioned algorithms at the stage of transforming the received data cube from a three-dimensional matrix into a vector, and also at the stage of computing the weight vector. The above tasks are presented in the figure below [9-11].

The article presents only the course of operation of the SMI STAP algorithm along with the marked stages that are the subject to parallel calculations. However, exactly the same steps are computed in parallel to the DDLS STAP method, hence one drawing is sufficient.

One of many possibilities offered by the MATLAB environment is to perform parallel computations using the parfor loop from the PCT package. Parfor is a way to run for loops in parallel, similar to OpenMP. Moreover, when using the parfor command, MATLAB automatically determines which variables are to be shared, private, or are reducing variables based on the form of your loop. All these facts must be recognised and declared in OpenMP.

The parfor statement changes the way a program performs calculations. It ensures that all iterations of a loop are independent, and they can be executed in any order or in parallel.

Execution starts with a single CPU (*Central Processing Unit*), the client. When a parfor loop is encountered, the client is helped by a "pool" of employees (threads). Each employee is assigned several iterations of the loop.

After the loop completes, the client resumes control of the execution. MATLAB ensures that the results are the same whether the program is executed sequentially or with the help of employees. The user can wait pending execution to determine how many employees are actually available.

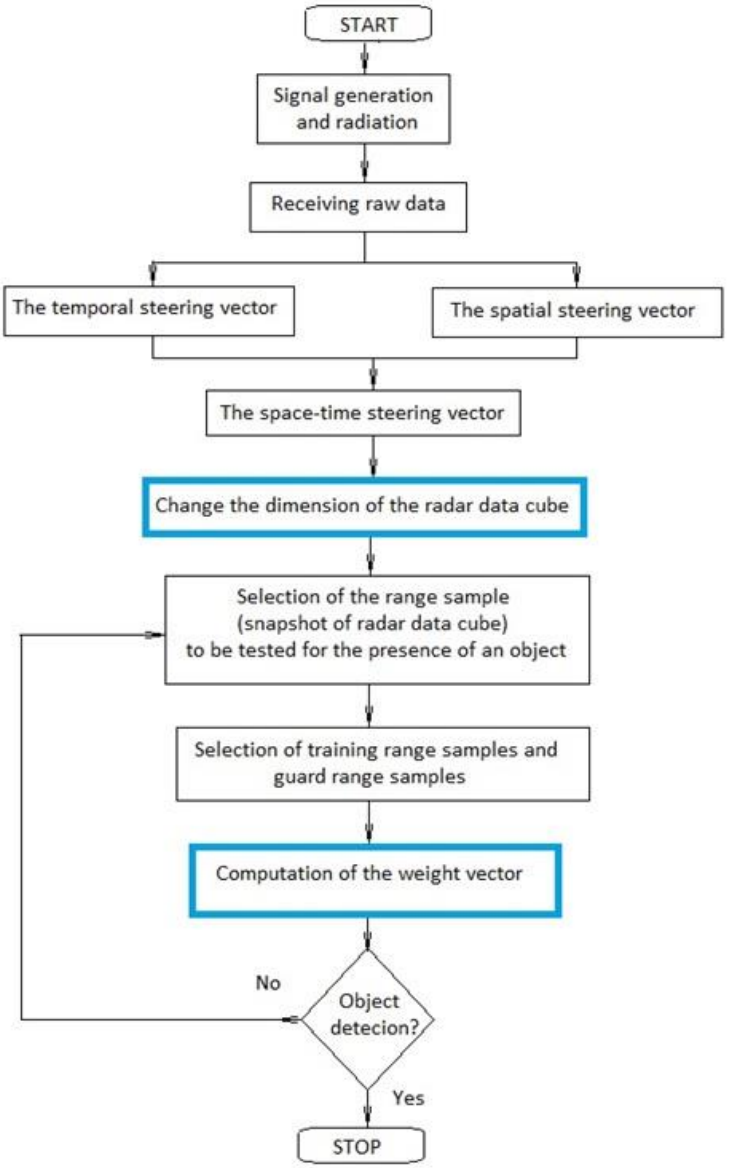

Fig. 3. Application of parallel programming at individual stages of SMI STAP

There are limitations to be aware of when using the parfor loop. These are among others:

- scalar variables defined outside a parfor loop but used inside are broadcasted to all workers,
- only the relevant parts of array variables defined outside a parfor loop but used inside are broadcasted to all workers,
- the parfor loop index can be used inside the parfor loop,
- the parfor loop index must take on integer values,
- parfor loops cannot be nested,
- all iteration-dependent behaviour is prohibited,
- load-balancing is not accounted for.

Briefly, parfor is usable when vectors and arrays that are modified in the calculation can be divided up into distinct slices, so that each slice is only needed for one iteration.

4. SIMULATIONS AND DISCUSSION

In order to compare the accuracy of object detection and the efficiency of both proposed methods, a series of simulations of a given radar system installed on board an aircraft detecting an object on the ground surface was performed. The simulations were made in the MATLAB environment. The parameters specified in the table below were assumed for the simulation. Our experiments are designed on the stand-alone with 2.30 GHz Intel Core i5 CPU, 4 GB memory size and 2-cores processor.

The objectives of parallelisation are defined in the literature on the subject. These are shortening the running time of programs, increasing the efficiency of calculations and obtaining faster processing. The computational efficiency is expressed by relative measures in which the program execution times (task solving) are compared on one and many processors. In the case of this article, performance will be expressed in a relative measure where the program execution time is compared sequentially and over several threads. The best-known measure of computing efficiency is speedup, expressed by the formula:

$$S_{\mathrm{p}} = \frac{T_1}{T_{\mathrm{p}}} \tag{14}$$

where p is the number of threads (processors), T_1 is the execution time of the sequential algorithm, and T_{p} is the execution time of the parallel algorithm using p threads (processors).

The definition of the speedup parameter has evolved over the years. First, in 1967, Amdahl's law stated that increasing the speed of a program when using multiple processors in parallel computing is limited by the time it takes to sequentially divide the program.

This was partially negative for the study of parallel computer architectures, and it played a pessimistic role. Then in 1987 Gustafson's Law indicated that any big enough problem can be effectively paralleled. Gustafson's law refers to the drawback of Amdahl's law, which is not scalable enough to take into account the availability of computing power as the machine grows. It addresses the problem of a fixed problem size or fixed computation loading on parallel processors. Instead, it proposes a fixed-time concept that leads to scaled speedup. Later, in 1990, Sun-Ni summarised Amdahl's law and Gustafson's law and he presented the formula for memory-limited acceleration - the Sun-Ni law. Since the Sun-Ni law is not constrained by constant load or time, it is more flexible and expressible system acceleration [14-16].

Table 2. Parameters assumed during the simulation

Parameter	**Value**
Radar operating frequency	10 GHz
Pulse repetition frequency	30 kHz
Sampling frequency	6 MHz, 3 MHz, 1.5 MHz
Duration of the pulse	0.33 μs
Inter-element spacing of the antenna	0.015 m
Radar position	[0; 0; 1000 m]
Object position	[1000 m; 1500 m; 0]
Jammer position	[1000 m; 1500 m; 1000 m]
Platform velocity	[0; 250 m/s; 0]
Object velocity	[30 m/s; 30 m/s; 0]

To reflect the real use of STAP processing in the airborne radar, a constant number of antennas in the antenna array was assumed for the simulation.

The analysis of the influence of the proposed solution on the duration of the calculations was carried out by adjusting the number of transmitted pulses and their sampling frequency. This had a direct effect on the number of range samples and, more importantly, on the size of the radar data cube. The parallelisation of the computations was obtained by dividing the partial computations in the program loops into four threads. From Tables 3 and 4, it can be concluded that the use of software parallelising loops for calculations is important, because the time needed to determine the position of the object decreases. Moreover, it is evident that the increase in the amount of processed data results in an extended computation time.

Table 3. Parameters assumed during the simulation and the obtained results for the SMI method

Dim.	Antennas	Pulses	Range Samples	SMI time [ms]	SMI parallel time [ms]	Speedup
1	10	10	50	0.2041	0.1576	1.295
2	10	20	50	0.3005	0.2677	1.123
3	10	30	50	0.4013	0.3725	1.077
4	10	10	100	0.2640	0.2526	1.045
5	10	20	100	0.4664	0.3800	1.227
6	10	30	100	0.6629	0.5617	1.180
7	10	10	200	0.5652	0.3431	1.647
8	10	20	200	0.8210	0.4944	1.661
9	10	30	200	1.2630	0.6241	2.024

Table 4. Parameters assumed during the simulation and the obtained results for the DDLS method

Dim.	Antennas	Pulses	Range Samples	SMI time [ms]	SMI parallel time [ms]	Speedup
1	10	10	50	0.0680	0.0525	1.295
2	10	20	50	0.1002	0.0892	1.123
3	10	30	50	0.1338	0.1242	1.077
4	10	10	100	0.0910	0.0842	1.081
5	10	20	100	0.1608	0.1267	1.269
6	10	30	100	0.2286	0.1872	1.221
7	10	10	200	0.1884	0.1107	1.702
8	10	20	200	0.2737	0.1595	1.716
9	10	30	200	0.4210	0.2013	2.091

Figure 4 shows the dependence of speedup on the dimension of the radar data cube. It can be seen an almost identical course of the curve for both methods. Moreover, it should be emphasised that with the increase in the dimension of the radar data cube, the use of the parfor loop to parallelise the calculations becomes more and more important, which of course results in higher speedup values. The conclusion from this study is that parallel computations should be applied to large matrices. For a smaller amount of data this is pointless as the speedup value is 1.

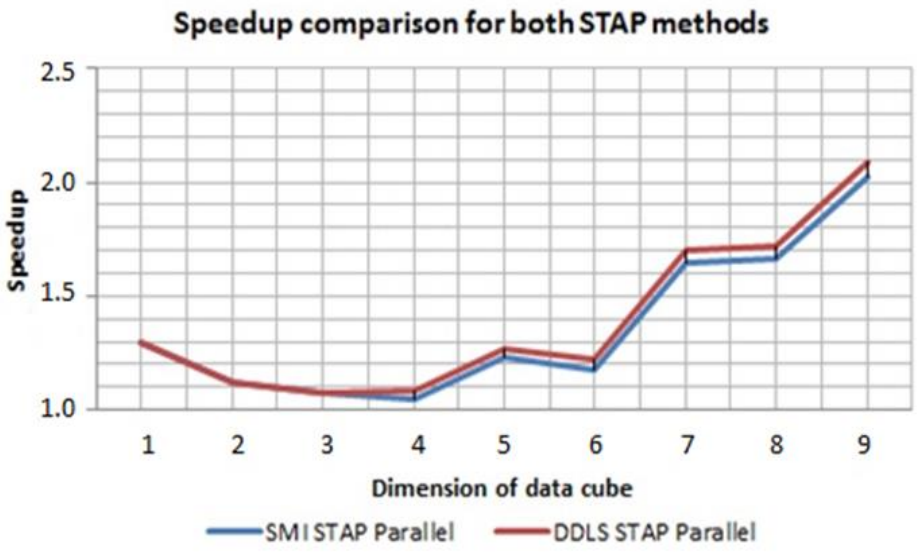

Fig. 4. Comparison of the speedup for the SMI method and the DDLS method using parallel programming

The second, key analysis of the effects of the proposed changes in the STAP algorithms was to check the accuracy of the object detection.

The graphs of the signal amplitude values are shown in the figures above from in-dividual range samples after STAP processing for the case when the number of range samples was equal to 200. The analysis of the obtained results shows that all four methods unambiguously detected the object with similar accuracy.

Performance evaluation is a general concept, which means that the correct detection of an object against an interference background - the highest value of the received signal, after processing by the selected algorithm - is considered reasonable and consistent with the assumptions made. In this paper, the attention and main effort has been focused on the operation of computational parallelisation. Therefore, a series of Monte Carlo simulations of individual algorithms were not performed to determine the magnitude of classical parameters used in radiolocation such as false alarm probability or correct detection probability.

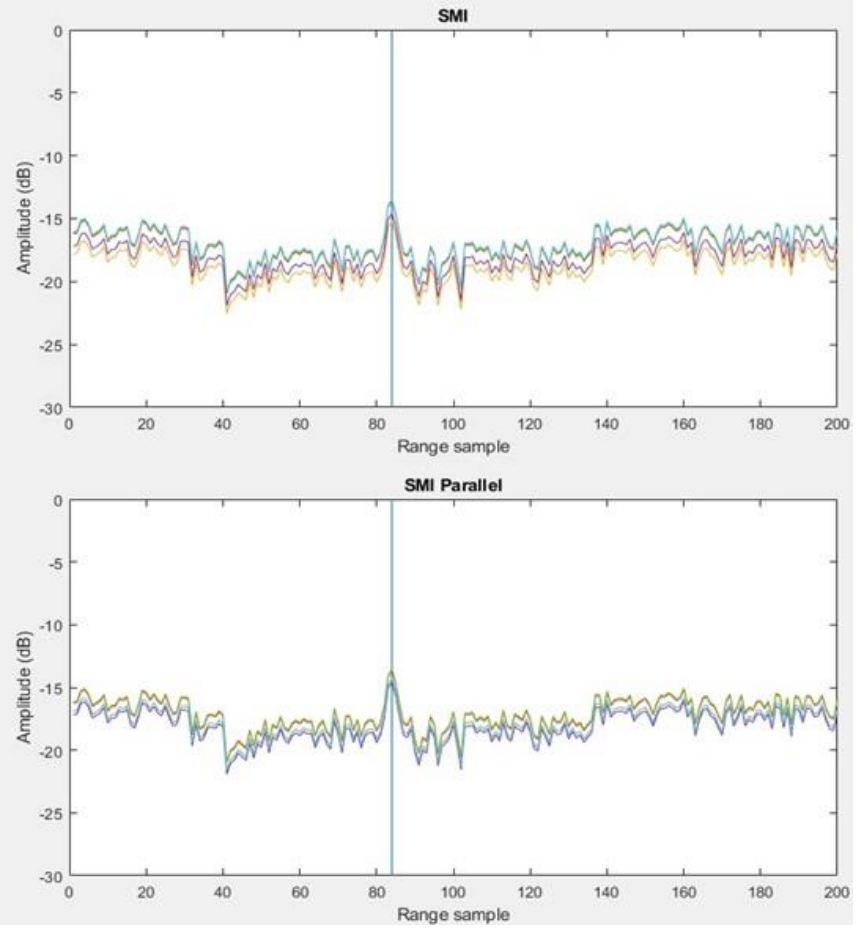

Fig. 5. Comparison of the detection performance of the SMI method and the SMI method using parallel programming (SMI Parallel)

Hence, the next step of our research may be to determine parameters such as false alarm probability or correct detection probability of individual STAP algorithms.

STAP is designed to filter out echoes from interference sources, and preserve the signal coming from the object of interest. In the literature, radar interference is divided into passive interference (clutter) and active interference (jammer). The elements responsible for the existence of passive interference are stationary terrain obstacles, while the elements responsible for the existence of active interference are jamming devices that are the source of active interference. Both types of interference are defined in detail in the Matlab environment through built-in functions that describe the interference environment.

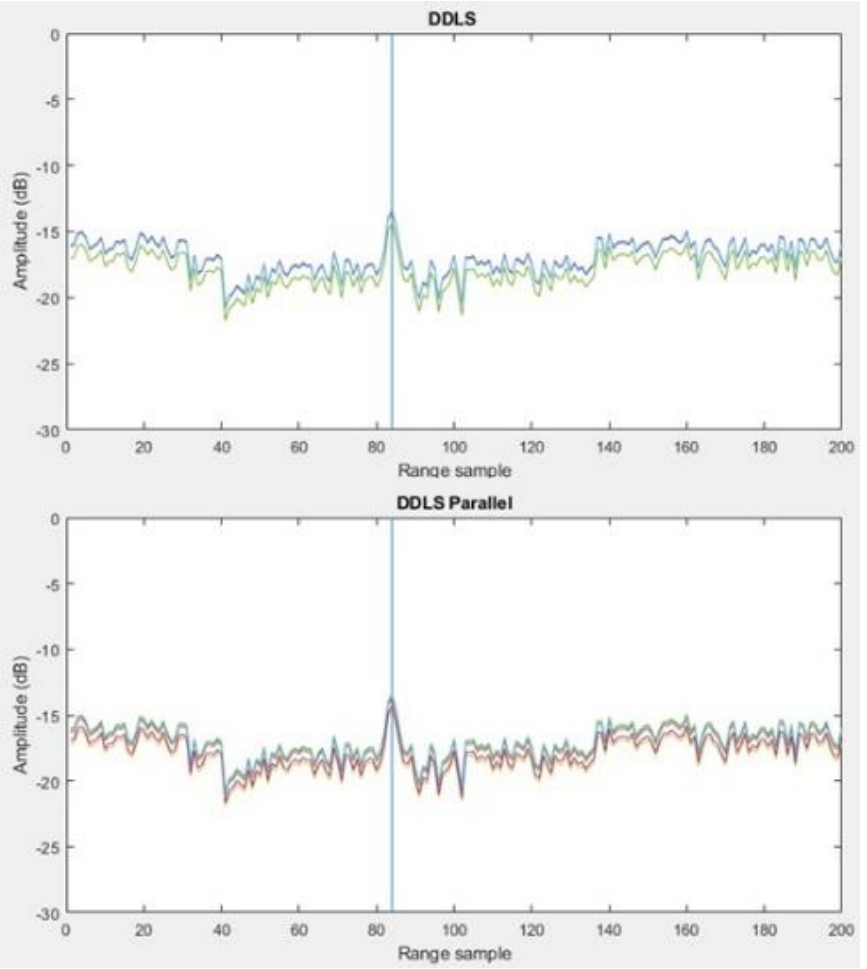

Fig. 6. Comparison of the detection performance of the DDLS method and the DDLS method using parallel programming (DDLS Parallel)

Of course, for STAP based on the statistical method of estimating the interference covariance matrix, the detection of the object occurs after searching a sufficiently large number of training samples, according to the algorithms described above.

5. CONCLSIONS

This article presents the description and implementation of two known STAP algorithms - SMI and DDLS. In addition, the article attempts to use parallel programming in the above methods to reduce the time of expensive calculations on very large matrices.

The duration of STAP calculations for four algorithms was analysed and compared. When analysing the obtained results, one should notice significantly shorter data processing times in the case of dividing the calculations into four threads. Due to the broadness of the obtained results and their scalability, the authors limited the presentation of the results only to the case when the number of range samples was equal to 200. Based on the simulations performed, it was found that the use of parallel programming requires additional time for communication and synchronisation of partial calculations from each thread.

The common feature of both SMI and DDLS methods is the performance of a series of mathematical operations, described in Table 1, for each hypothetical object. However, the differences between the two methods in favour of DDLS are lower computational complexity and shorter data processing time.

It is true that MATLAB is not the best tool for parallel computing. However, it is the only initial platform available to authors on which solutions have been tested. Where hardware conditions permit, further testing can be performed on other platforms such as CUDA. A very important conclusion from the research is that the parfor structure can only be used for parallel execution of independent iterations of the *for* loop.

The article presents the possibility of a parallel application for the STAP algorithm based on the statistical method of estimating the interference covariance matrix.

In addition, the cited literature on the subject also discusses the use of, for example, CUDA architecture in algorithms based on the statistical method of estimating the interference covariance matrix. Hence, the idea is born to implement the STAP algorithm in the MATLAB environment, based on the non-statistical method of estimating the interference covariance matrix, which the authors studied earlier [20].

The idea of using parallel programming in STAP, supported only by the preliminary research results presented above, gives a real chance for the casual implementation of the STAP algorithm in a radar operating in close to real time.

FUNDING

The authors received no financial support for the research, authorship, and/or publication of this article.

REFERENCES

[1] Klemm, Richard. 2002. *Principles of space-time adaptive processing*. IEE Publishing.
[2] Skolnik, Merril. 2001. *Introduction to Radar Systems*. New York: The McGraw-Hill Companies.

[3] Guerci, Joseph. 2003. *Space – Time Adaptive Processing for Radar*. Artech House Publishers.
[4] Ward, James. 1994. *Space-time adaptive processing for airborne radar*. Technical Report No. 1015, MIT Lincoln Laboratory.
[5] Melvin, William. 2004. „A STAP overview". *IEEE Transactions on Aerospace and Electronics Systems Magazine* 19 (1) : 19-53.
[6] Le Caillec, Jean-Marc, Tomasz Górski, Guillaume Sicot and Adam Kawalec. 2018. „Theoretical Performance of Space-Time Adaptive Processing for Ship Detection by High-Frequency Surface Wave Radars". *IEEE Journal of Oceanic Engineering* 43 (1) : 238-257.
[7] Górski, Tomasz. 2008. *Space – Time Adaptive Signal Processing for Sea Surveillance Radars*. Doctoral dissertation. Warsaw, Poland: Military University of Technology.
[8] Tao, Su and Bao Zheng. 2001. Parallel Processor in Space Time Adaptive Signal Processing. *Proceedings CIE International Conference on Radar*.
[9] Lebak, James and Adam Bojanczyk. 2000. „Design and performance evaluation of a portable parallel library for space-time adaptive processing". *IEEE Transactions on Parallel and Distributed Systems* 11 (3) : 287-298.
[10] Rajan, Kanhirodan and Lalit Patnaik. 2003. „Implementation of STAP algorithms on IBM SP2 and on ADSP 21062 dual digital signal processor systems". *Microprocessors and Microsystems* 27 (4) : 221-227.
[11] Shao, Yin-Bo, Yong Wang, Yu Dend and Qiang Li. 2006. The Universal Implementation of Space-Time Adaptive Processing. In *Proceedings of the CIE International Conference on Radar*.
[12] Gawande, Nitin, Joseph Manzano, Antonio Tumeo, Nathan Tallent, Darren Kerbyson and Adolfy Hoisie. 2015. Power and performance trade-offs for Space Time Adaptive Processing. In *Proceedings of the IEEE 26th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*.
[13] Mahmoudi, Sidi, Michał Kierzynka, Pierre Manneback and Krzysztof Kurowski. 2015. „Real-time motion tracking using optical flow on multiple GPUs". *Bulletin of the Polish Academy of Sciences. Technical Sciences* 61 (4) : 989-992.
[14] Szczepankiewicz, Karolina, Mateusz Malanowski and Michał Szczepankiewicz. 2015. „Passive Radar Parallel Processing Using General-Purpose Computing on Graphics Processing Units". *Bulletin of the Polish Academy of Sciences. Technical Sciences* 61 (4) : 357-363.
[15] Kaushik, Mayank, Joachim Trinkle and Joe Fabrizio. 2019. A computation and memory efficient implementation of STAP for an airborne side-looking radar. In *Proceedings of the International Radar Conference (RADAR)*.
[16] Chaparro, Luis and Aydin Akan. 2019. *Signals and systems using MATLAB*. London, UK: Academic Press.

[17] Zaid, Alyasseri. 2015. *Introduction to Parallel Computing using Matlab*. Lambert Academic Publishing.
[18] Kepner, Jeremy. 2009. *Parallel MATLAB for Multicore and Multinode Computers*. Society for Industrial and Applied Mathematics.
[19] Amdahl, Gene. 1967. Validity of the single-processor approach to achieving large scale computing capabilities. In *Proceedings of the AFIPS Conference*.
[20] Gustafson, John. 1988. „Reevaluating Amdahl's law". *Communications of the ACM* 31 (5) : 532-533.
[21] Sun, Xian and Lionel Ni. 1990. Another view on parallel speedup. In *Proceedings of the IEEE Supercomputing '90*.
[22] John, Mathew, Michael Inggs and Dario Petri. 2011. „Real Time Processing of Networked Passive Coherent Location Radar System". *International Journal of Electronics and Telecommunications* 57 (3) : 363-368.
[23] Ślesicka, Anna and Adam Kawalec. 2020. „An Application of the Orthogonal Matching Pursuit Algorithm in Space-Time Adaptive Processing". *Sensors.* 20 (12) : 3468-1-13.

Badanie możliwości zastosowania programowania równoległego do algorytmów adaptacyjnego przetwarzania przestrzenno-czasowego

Błażej ŚLESICKI[1], Adam KAWALEC[2], Anna ŚLESICKA[2]

[1]*Lotnicza Akademia Wojskowa, ul. Dywizjonu 303, Dęblin*
[2]*Wojskowa Akademia Techniczna, Wydział Mechatroniki Uzbrojenia i Lotnictwa ul. gen. Sylwestra Kaliskiego 2, 00-908 Warszawa*

Streszczenie. W artykule przedstawiono opis, założenia i kolejne kroki algorytmów przestrzenno-czasowego przetwarzania adaptacyjnego (STAP) stosowanych jako narzędzie przetwarzania sygnałów w radarach. Porównano i pokazano możliwości wykrywania obiektów za pomocą algorytmów Sample Matrix Inversion (SMI) i Data Domain Least Squares (DDLS). W artykule przedstawiono wpływ zastosowania programowania równoległego na czas obliczeń obu algorytmów. Głównym celem pracy było zaproponowanie efektywnej metody implementacji algorytmu STAP w czasie rzeczywistym w pokładowych systemach radarowych.
Słowa kluczowe: przestrzenno-czasowe adaptacyjne przetwarzanie sygnałów, przetwarzanie sygnałów radarowych, radar