Досліджено проблему недостатнього розвитку алгоритмів, реалізованих у мікропрограмному забезпеченні для пошуку точок перетину квадрик. Розроблено, реалізовано та досліджено алгоритм локалізації точок перетину квадрик на основі властивостей неперервних диференційовних функцій у замкнутій області. Новим алгоритмом досягається вища релевантність результатів, ніж його єдиним аналогом. Результати призначені для інтелектуальних сенсорів векторних величин

Ключові слова: інтелектуальні сенсори векторних величин, локалізація точок перетину квадрик, мікроконтролер ARM

Исследована проблема недостаточного развития алгоритмов, применимых для реализации в микропрограммном обеспечении для поиска точек пересечения квадрик. Разработан, реализован и изучен алгоритм локализации точек пересечения квадрик на основе свойств непрерывных дифференцируемых функций в замкнутой области. Новым алгоритмом достигается релевантность результатов выше, чем у его единственного аналога. Результаты предназначены для интеллектуальных сенсоров

Ключевые слова: интеллектуальные сенсоры векторных величин, локализация точек пересечения квадрик, микроконтроллер ARM

UDC 004.4, 519.688
DOI: 10.15587/1729-4061.2016.84889

DEVELOPING AN ALGORITHM WITH IMPROVED RELEVANCE FOR THE LOCALIZATION OF VECTOR'S COORDINATES FOR INTELLIGENT SENSORS

T. Marusenkova
PhD, Senior lecturer
Department of software
Lviv Polytechnic National University
S. Bandery str., 12, Lviv, Ukraine, 79013
E-mail: tetyana.marus@gmail.com

1. Introduction

Sensors of vector quantities are an important link in the collection of primary data for their further processing. In order to increase sensitivity of the sensors and, thus, the accuracy of measurement, as well as to reduce their dimensions and, therefore, to increase density of mapping of the measured vector quantity, new constructive solutions for the sensors of vector quantities are being developed [1].

However, improving some indicators often entails the worsening of others. As a result, there are several sensors of vector quantities of a new generation whose field characteristics depend on all three coordinates of the measured vector quantity, their squares and pairwise products [2]. Determining coordinates of the sought vector comes down in this case to solving a system of equations that describe surfaces of the second order (quadrics). Such a complicated procedure of processing measurement results leads to the idea of building intelligent sensors of vector quantities, that is, for the convenience of using sensors with non-linear characteristics, they are added with a "brain", capable of performing the required calculations [3–5]. A microcontroller that is inferior to the computer in clock frequency and the volume of permanent and operative memory often performs a role of the "brain". Accordingly, the software of intelligent sensors, as well as of any embedded systems, should meet the requirements of economic usage of memory of programs and data, as well of reduced computational complexity.

Since analytical solutions can be obtained only for the narrow classes of systems of equations, search for the solutions of equations system implies localization of the regions where they are located, and their further refinement. Classic search algorithms for the solutions of systems of equations of quadrics are not applicable for the implementation in the software of intelligent sensors due to their sensitivity to accuracy of preserving results of the intermediate calculations. The existing algorithms, focused on the realization in the microprogramming software, are not sufficiently elaborated, and this fact necessitates conducting research aimed at improving the performance indicators of these algorithms, first of all, accuracy, computational complexity and the volumes of applied permanent and operative memory.

2. Literature review and problem statement

One of the most powerful approaches to the solution of polynomial equations and their systems is the application of a Gröbner basis. In 1965 the algorithm was proposed that makes it possible in only a finite number of steps to build a Gröbner basis of the ideal [6]. The advantage of the Buchberger's algorithm is universality – the algorithm is applicable for any system of polynomial equations. However, the Buchberger's algorithm is oriented towards symbolic computations and it is quite slow for large systems, which limits its applicability in practice. A number of scientists conducted studies of alternative, accelerated ways of constructing a Gröbner basis. In 1883 an alternative algorithm for constructing a Gröbner basis was proposed, faster than Buchberger's algorithm. Subsequently, the concept of this algorithm formed the basis of the F4 algorithm (1994), which made it possible to avoid redundant intermediate calculations and was subsequently implemented in Maple. In 2002 the F5 algorithm was proposed, which is considered

to be one of the fastest algorithms for constructing a Gr bner basis [7]. The F5 algorithm uses the sparse matrices, which, as is known, take up a lot of memory. The F5C algorithm, accelerated modification of the F5 algorithm, is proposed in [8]. Further improvement of the algorithm is achieved in [9]. In [10] proposed G2V – the next modification of F5. In parallel with the new algorithms, there were also conducted parallel computations of the classic Buchberger's algorithm. An analysis of recent publications dealing with the development of applying a Gröbner basis for solving polynomial equations and their systems indicates that there are studies in the direction of reducing the computational complexity of algorithms for its construction, however, without losing the universality of these algorithms. However, the newest modifications of algorithms for constructing a Gröbner basis are still not applicable in microprogramming software due to their complexity, requirements to the accuracy of saving intermediate results of transformations and required memory. On the other hand, the application of these universal algorithms is excessive when it is necessary to solve only a system of three equations of surfaces of the second order. Thus, the implementation of the algorithms for constructing a Gröbner basis is contrary to the general principles of minimal consumption of resources in the software of the embedded systems.

A search for and mapping of the intersection of surfaces of the second order is a task that has gained importance with the development of computer graphics and computer aided design systems. In the latter, a capacity to process curves and surfaces appeared in the 1960s; however, the techniques embedded in them lacked either accuracy or acceptable time of realization. A number of publications are devoted to the problem of accurate calculation of the curves of intersection of surfaces and the points of intersection of curves. Mathematical basis was designed for computing the ordering of arbitrary quadrics (in particular, degenerated) in a three-dimensional space. To do this, the problem is reduced to the lesser dimension, that is, to the two-dimensional case, which is solved by cylindrical algebraic decomposition. An implementation of the latter requires a lot of computing in itself and there has recently been the work carried out to simplify this process, in particular, by means of a Gröbner basis [11]. In [12] solved the problem of efficient and accurate parametric representation of the curve of intersection of two quadrics. These results led to further development in [13], in which an effective, accurate and complete algorithm was proposed for constructing the graph of contiguity of ordering quadrics.

However, an analysis of existing publications in the field of computational geometry reveals that the attention of scientists is focused on the development of symbol-numerical algorithms that are inexpedient to use in micro programming software.

There are libraries, optimized for different architectures of microcontrollers that implement numerical methods for solving arbitrary equations, regardless of their nature. As is known, the solution of nonlinear equations by numerical methods include separation of roots (localization of solutions) and their refinement. [14] examined influence of the localization of potential solutions of a system of nonlinear algebraic equations on the success of refinement of these solutions by library functions, applicable for microcontrollers of the ARM architecture, that implement the known numerical methods (as a rule, this is a method of dichotomy, the Newton's method and the method of chords). All library functions require an approximate solution, which in the form of numbers or intervals is passed to them as parameters. In the course of study of the specified library functions, it was found that their work is unstable if, as the parameter, an incorrectly formed interval was transmitted (not narrow enough or the one that contains several solutions of the equation) or not sufficiently approximated solution. A multi-dimensional Newton's method allows specification of the approximate solution of the system of nonlinear equations; however, for any initial vector of initial approximations, the method is able to find only one solution, therefore, in order to search for all the solutions, it is necessary to know their number and regions, in which they are located. All this testifies to the importance of proper localization of the solutions. In an ideal case, as a result of the stage of localization of solutions of equations system, there has to be found as many regions as there are solutions.

However, at present there are no ready libraries for the localization of solutions of polynomial equations and their systems. Thus, a relevant task is the development of algorithms and their implementations for the localization of solutions of polynomial equations and their systems, which would meet the requirements to the minimum resource consumption. Moreover, in parallel with the research into development of universal algorithms, there is a need for research aimed at developing quick and simple algorithms that would allow us at minimum expenses to solve a certain narrow class of problems.

[15] presents the algorithm for the separation of roots of polynomial functions of one variable and microprogramming software for the ARM family of controllers. [16] described a method for the localization of solutions of systems of equations of the form

$$a_{11}x^2 + a_{22}y^2 + a_{33}z^2 + a_{14}x + a_{24}y + a_{34}z + a_{44} = 0$$

using Sylvester's matrix calculation. [14] presented an algorithm for the localization of polynomial equations and their systems, which uses interval arithmetic. The basic idea of the algorithm is as follows. All the equations in the system are reduced to the form so that their right parts are equal to 0. All the region of the search for solutions of the system of equations is divided into contiguous subregions in the form of the same size rectangular parallelepipeds

$$x_i \le x \le x_{i+1},\ \ y_j \le y \le y_{j+1},\ \ z_k \le z \le z_{k+1}.$$

Instead of the point values of x, y and z, each of the functions that represent the left parts of the equation, is substituted with intervals $[x_i, x_{i+1}]$, $[y_j, y_{j+1}]$, $[z_k, z_{k+1}]$. As a result, we obtain three intervals. If the interval contains 0, then there is a point in the examined region, in which function takes a 0 value, that is, this point is a solution of the corresponding equation in the system. Otherwise, there is no such a point in this region. Thus, if all the calculated intervals include 0, we consider this region to potentially contain solutions. Otherwise, the region is neglected. The algorithm is simple to implement and is guaranteed to find all regions containing solutions, but its shortcoming is that part of the found regions is a "false alarm", that is, it actually contains no points of quadrics intersection. Therefore, there is a need to search for alternative algorithms that would provide a higher relevancy of results.

3. The aim and tasks of the study

The aim of this work is to improve the algorithm for the localization of solutions of the systems of equations of quadrics for intelligent sensors of vector quantities based on microcontrollers of the ARM Cortex-M architecture by relevance of the found regions with potential solutions. If the speed, permanent and operative memory usage of the algorithm, improved by this indicator, worsens, then such worsening should be compensated for by the increase in relevance of the found regions.

To achieve the set aim, the following tasks were formulated:

– to analyze and systematize mathematical apparatus that can serve the foundation for improving the algorithm of localization of the points of quadrics intersection, to develop and implement a new algorithm for a microcontroller based on the ARM Cortex-M4 architecture;

– to conduct a comparison of implementations of the existing and improved algorithms for the localization of the points of quadrics intersection by relevance of the regions, found by these algorithms, with potential points of intersection, as well as the time of execution, the amount of code and the usage of operative memory.

4. 1. Mathematical apparatus for the localization of points of quadrics intersection

Let an intelligent sensor of vector quantity contains three detectors whose field characteristics are described by quadric equations:

$$\begin{aligned}
&a_{11}x^2+a_{12}y^2+a_{13}z^2+b_{11}xy+b_{12}xz+\\
&+b_{13}yz+c_{11}x+c_{12}y+c_{13}z-S_1=0,\\
&a_{21}x^2+a_{22}y^2+a_{23}z^2+b_{21}xy+b_{22}xz+\\
&+b_{23}yz+c_{21}x+c_{22}y+c_{23}z-S_2=0,\\
&a_{31}x^2+a_{32}y^2+a_{33}z^2+b_{31}xy+b_{32}xz+\\
&+b_{33}yz+c_{31}x+c_{32}y+c_{33}z-S_3=0,
\end{aligned} \qquad (1)$$

where x, y and z are the projections of the measured vector quantity on the vertical axis, horizontal axis and applicate of certain Cartesian coordinate system, bound to the sensor body, a_{ij}, b_{ij}, c_{ij} $(i=\overline{1,3},\ j=\overline{1,3})$ are the known coefficients of the field characteristics of detectors, S_i are the measured signals of each of the three detectors.

Let us denote through f_1, f_2 and f_3 those functions that are the left parts of equations (1):

$$\begin{aligned}
&f_1=a_{11}x^2+a_{12}y^2+a_{13}z^2+b_{11}xy+\\
&+b_{12}xz+b_{13}yz+c_{11}x+c_{12}y+c_{13}z-S_1,\\
&f_2=a_{21}x^2+a_{22}y^2+a_{23}z^2+b_{21}xy+\\
&+b_{22}xz+b_{23}yz+c_{21}x+c_{22}y+c_{23}z-S_2,\\
&f_3=a_{31}x^2+a_{32}y^2+a_{33}z^2+b_{31}xy+\\
&+b_{32}xz+b_{33}yz+c_{31}x+c_{32}y+c_{33}z-S_3.
\end{aligned} \qquad (2)$$

Let us consider in detail the behavior of function f_1 within the rectangular parallelepiped $x_a\le x\le x_b$, $y_a\le y\le y_b$, $z_a\le z\le z_b$. If at different tops of the parallelepiped the function takes different signs, then, due to its continuity, there will be such a point within the specified rectangular parallelepiped in which function takes the value 0. But, if, at all tops of the parallelepiped, the function either positive or negative, we cannot conclude that in the given region there is no such a point where the function becomes zero, additional tests are required.

As is known, function acquires the maximum (and minimum) value in a limited region either in the points of extremum or on the border. Thus, it is necessary to find the maximum value on the border and compare it with the values in the points of extremum inside the region if there are such points there.

A border of parallelepiped is six of its edges. On each of these edges, one of the variables of function f_1 is fixed, that is, we receive 6 functions from two variables that take the value 0 in the critical points:

$$\begin{aligned}
&a_{11}x^2+a_{12}y^2+a_{13}z_a^2+b_{11}xy+b_{12}xz_a+\\
&+b_{13}yz_a+c_{11}x+c_{12}y+c_{13}z_a-S_1=0,\\
&a_{11}x^2+a_{12}y^2+a_{13}z_b^2+b_{11}xy+b_{12}xz_b+\\
&+b_{13}yz_b+c_{11}x+c_{12}y+c_{13}z_b-S_1=0,\\
&a_{11}x^2+a_{12}y_a^2+a_{13}z^2+b_{11}xy_a+b_{12}xz+\\
&+b_{13}y_az+c_{11}x+c_{12}y_a+c_{13}z-S_1=0,\\
&a_{11}x^2+a_{12}y_b^2+a_{13}z^2+b_{11}xy_b+b_{12}xz+\\
&+b_{13}y_bz+c_{11}x+c_{12}y_b+c_{13}z-S_1=0,\\
&a_{11}x_a^2+a_{12}y^2+a_{13}z^2+b_{11}x_ay+b_{12}x_az+\\
&+b_{13}yz+c_{11}x_a+c_{12}y+c_{13}z-S_1=0,\\
&a_{11}x_b^2+a_{12}y^2+a_{13}z^2+b_{11}x_by+b_{12}x_bz+\\
&+b_{13}yz+c_{11}x_b+c_{12}y+c_{13}z-S_1=0.
\end{aligned} \qquad (3)$$

Each of the six functions that represent the left side of equations (3), in turn, takes maximum value either in critical points (if they are in the region, limited by the appropriate rectangle) or at the borders.

The limits for functions that represent the left side of equations (3) are the edges of the rectangular parallelepiped $x_a\le x\le x_b$, $y_a\le y\le y_b$, $z_a\le z\le z_b$:

$$\begin{aligned}
&a_{11}x^2+a_{12}y_i^2+a_{13}z_j^2+b_{11}xy_i+b_{12}xz_j+\\
&+b_{13}y_iz_j+c_{11}x+c_{12}y_i+c_{13}z_j-S_1=0,\\
&a_{11}x_k^2+a_{12}y^2+a_{13}z_j^2+b_{11}x_ky+b_{12}x_kz_j+\\
&+b_{13}yz_j+c_{11}x_k+c_{12}y+c_{13}z_j-S_1=0,\\
&a_{11}x_k^2+a_{12}y_i^2+a_{13}z^2+b_{11}x_ky_i+b_{12}x_kz+\\
&+b_{13}y_iz+c_{11}x_k+c_{12}y_i+c_{13}z-S_1=0,
\end{aligned} \qquad (4)$$

where $x_k\in\{x_a,x_b\}$, $y_i\in\{y_a,y_b\}$, $z_j\in\{z_a,z_b\}$ (that is, for each equation there are four combinations of pairs of fixed variables).

Each of the functions that represent the left part of equations (4) takes the maximum (and minimum) value either in critical points (if they exist on the corresponding face of the examined parallelepipeds) or at the ends of the section. The ends of the section are the tops of the parallelepiped represented by all points (x_k,y_i,z_j), $x_k\in\{x_a,x_b\}$, $y_i\in\{y_a,y_b\}$, $z_j\in\{z_a,z_b\}$.

To find critical points of each of the functions that represent the left part of equations (3), we find partial derivatives and equal them to zero. For example, for the first of these functions:

$$a_{11}x^2 + a_{12}y^2 + a_{13}z_a^2 + b_{11}xy + b_{12}xz_a + \\ + b_{13}yz_a + c_{11}x + c_{12}y + c_{13}z_a - S_1 = 0$$

critical points are determined from the system of linear algebraic equations:

$$2a_{11}x + b_{11}y + b_{12}z_a + c_{11} = 0,$$

$$b_{11}x + 2a_{12}y + b_{13}z_a + c_{11} = 0. \qquad (5)$$

Critical points of the rest of the functions that represent the left side of equations (3) are determined in a similar way.

To find the extremum points of functions that represent the left side of equations (4), we will take their derivatives and equal them to zero:

$$2a_{11}x + b_{11}y_i + b_{12}z_j + c_{11} = 0,$$

$$2a_{12}y + b_{11}x_k + b_{13}z_j + c_{12} = 0,$$

$$2a_{13}z + b_{12}x_k + b_{13}y_i + c_{13} = 0. \qquad (6)$$

Thus, each of the functions that represent the left side of equations (4) may have not more than one critical point (if it belongs to this examined face) and critical points are determined by formulas:

$$x = -\frac{\left(b_{11}y_i + b_{12}z_j + c_{11}\right)}{2a_{11}},$$

$$y = -\frac{\left(b_{11}x_k + b_{13}z_j + c_{12}\right)}{2a_{12}},$$

$$z = -\frac{\left(b_{12}x_k + b_{13}y_i + c_{13}\right)}{2a_{13}}. \qquad (7)$$

Thus, if it is necessary to verify the existence of points of quadrics intersection in a certain region that has the shape of rectangular parallelepiped $x \in [x_a, x_b]$, $y \in [y_a, y_b]$, $z \in [z_a, z_b]$, then it is sufficient to determine the signs of function (2) at all tops of the parallelepiped, functions that represent the left side of equations (4), in points (7), that is, on the edges of the parallelepiped, and functions that represent the left side of equations (3), in points that are the solutions of systems of linear algebraic equations of the form (5). If for any of the functions of group (2) we detect the change in sign (for example, the function is negative in all tops (x_k, y_i, z_j), $x_k \in \{x_a, x_b\}$, $y_i \in \{y_a, y_b\}$, $z_j \in \{z_a, z_b\}$, but it takes negative value in some of the points on the face of parallelepiped or its edge), then in the examined region there is a point, in which this function takes the value 0. If the change in signs is detected for each of the functions (2), there is a probability that in the examined region there is a point of quadrics intersection (1). It is not excluded that within this region all quadrics intersect in pairs, but not in one point, which is why the answer to the question about the existence of intersection point can be obtained only after the refinement of solutions by one of the known methods or when $x_b - x_a$, $y_b - y_a$ and $z_b - z_a$ are so small that in the middle of ranges $[x_a, x_b]$, $[y_a, y_b]$, $[z_a, z_b]$ can be considered with permissible error to be a solution of the system of equations (1). On the other hand, the absence of change in signs of at least one of the functions (2) indicates that in the examined region there are no points of quadrics intersection whatsoever (1).

4. 2. Procedure for determining the regions of quadrics intersection points

As in [14], the whole region D, in which we will search for solutions of the system of equations (2), we split in the same size rectangular parallelepipeds: take L of equidistant points on the Ox axis, M – on the Oy axis and N – on the Oz axis.

In contrast to the proposed by [14] approach to the localization of solutions of system of three equations of surfaces of the second order that is based on the interval arithmetic and implies the allocation of memory to save the calculated limits of intervals of the quadrics (1), ranges of variables $[x_k, x_{k+1}]$, $[y_i, y_{i+1}]$, $[z_j, z_{j+1}]$ and the intermediate results of computations, the approach presented in this paper, allows us to save only data on the signs of functions (2) in each of the node points and functions that represent the left parts of equations (3) and (4), in those their critical points, determined from (5) and (7), which belong to the edges and edges of the current examined rectangular parallelepiped. To keep the sign, one bit is sufficient (hereinafter 1 encodes a negative value, 0 – nonnegative) that allows us to design and implement localization algorithm for the points of quadrics intersection using variables of the unsigned integer type instead of variables of the valid type. In this case, in the designated variables every bit will be employed as opposed to the situation when in a variable part of bits is engaged with data, and the remaining bits are complementing zeros.

To keep the signs of function f_1, in node points we will create array F1_Nodes the size $M \times N$ of the unsigned integer type T that contains L bits. Each element F1_Nodes[i][j] of this array will represent the values of function f_1 in points, in which $y=y_i$, $z=z_j$, and x takes on value x_k $(k = \overline{1,L})$, and the bit with number k (in the LSB order) will correspond to the sign of function f_1 in point (x_k, y_i, z_j). For example, if M=N=8 and T is the type that is represented by 8 bits, then F1_Nodes[i][j]=10010011 will mean that function f_1 in points (x_1, y_i, z_j), (x_4, y_i, z_j), (x_7, y_i, z_j), (x_8, y_i, z_j) takes negative values and in points (x_2, y_i, z_j), (x_3, y_i, z_j), (x_5, y_i, z_j), (x_6, y_i, z_j) – nonnegative.

To save data on the signs of functions f_2 and f_3 in the same node points, we need two other arrays (F2_Nodes[i][j] and F3_Nodes[i][j]) of the same size and the same type. All three of these arrays will use $3L \times M \times N / 8$ bytes.

To track the existence of critical points and signs of derivative of each of three functions (2), let us create 6 arrays: F 1_PosExtremums, F1_NegExtremums, F2_PosExtremums, F2_NegExtremums, F3_PosExtremums and F3_NegExtremums. For each combination of x and y, x and z, and y and z, we find those common values of z, y and x, respectively, for which equations (4) are converted to equalities. Next, we determine to which ranges $[z_j, z_{j+1}]$ $(j = \overline{1,N-1})$ $(i = \overline{1,M-1})$ $[x_k, x_{k+1}]$ $(k = \overline{1,L-1})$ the found values of z, y and x, respectively, belong. It is enough to remember only the beginning of each of the specified ranges (because the length of each range is known). Each face (and critical point on it) may belong to four or two contiguous parallelepipeds or to only one parallelepiped. In the bit with number K of the element of array F1_PosExtremums[i][j] we will store 1, if at least on one of the 12 edges of parallelepiped, one of the tops of which is the point (x_k, y_i, z_j) and this particular top is closest to the origin of coordinates, there is a critical point where the function

that represents the left side of the first equation from the system of equations (4) takes a negative value. The purpose of array F1_NegExtremums is similar, the difference being that in it we save 1 in the case when function takes negative value in at least one critical point on the edges of parallelepiped with the top (x_k, y_i, z_j). Zeros in the bits of elements of these arrays will encode a lack of critical points in the corresponding parallelepipeds. Four other arrays are needed to save similar data on critical points of functions f_2 and f_3 on the edges of parallelepipeds.

Thus, when we find critical point (x, y_i, z_j), $x \in [x_k, x_{k+1}]$ of the function that represents the left side of the first equation of the system of equations (4), one should perform the following steps:

– assign the sign of function to it;

– in array F1_NegExtremums or F1_PosExtremums (depending on the sign of function), to assign the following bits to 1;

– bit with number k of the element with indexes i and j;

– bit with number k of the element with indexes (i–1) and j, if condition i>0 and i<M–1 is valid;

– bit with number k of the element with indexes i and (j–1), if condition j>0 and j<N–1 is valid;

– bit with number k of the element with indexes (i–1) and (j–1), if conditions (i>0 and i<M–1) and (j>0 and j<N–1) are valid.

For critical points that are located on the edges, parallel to the Oy axis and the Oz axis, the approach remains the same, only the analyzed indices change.

Next, one should determine the sign of maximum/minimum value of each of functions (2) on the edges of parallelepipeds. For this purpose, we will find the solution to the system of linear algebraic equations of the form (5) with two unknowns for each of functions (2) and each value of x_k, y_i and z_j ($k=\overline{1,L}$, $i=\overline{1,M}$, $j=\overline{1,N}$). Unlike critical points on the edges, for critical points on the sides it is necessary to determine the ranges, to which a pair of the found values belongs – y and z for planes $x=x_k$, x and z for planes $y=y_i$ and x and y for planes $z=z_j$.

Each side may belong only to one or two contiguous parallelepipeds. That is why, by analogy with critical points (7), we attribute the found critical points to the parallelepipeds, in each of which the top with the least coordinates is one of the node points. To keep the signs of functions on the sides of parallelepipeds, it is possible to use existing arrays, determined earlier for the signs of functions in critical points (7). If a critical point belongs to the parallelepiped for which point x_k, y_i and z_j ($k=\overline{1,L-1}$, $i=\overline{1,M-1}$, $j=\overline{1,N-1}$) is the top with the least coordinates, namely, its edges, perpendicular to the Ox axis, then the following steps should be taken:

– determine the sign of function in critical point;

– into one of the arrays (XX_PosExtremums or XX_NegExtremums), depending on the sign of function, assign the bits to 1;

– bit with number k of the element with indexes i and j;

– bit with number k of the element with indexes (i–1) and j, if condition i>0 and i<M–1 is valid.

For the planes, perpendicular to the Oy and Oz axes, the approach remains the same (only the indices change).

The total amount of memory for arrays F1_PosExtremums, F1_NegExtremums, F2_PosExtremums, F2_NegExtremums, F3_PosExtremums and F3_NegExtremums is $6L \times M \times N / 8$ bytes.

When all arrays are formed, it is possible to start to analyze the signs of functions in all node and critical points. To save the results, it is expedient to create array RootsLocalized of the same dimensions as that of each of the array defined previously. Each element of this array is determined as follows: we compare the signs of function f_1 in every top of parallelepiped (x_k, y_i, z_j) with the signs of the same function in the neighbouring tops (x_{k+1}, y_i, z_j), (x_k, y_{i+1}, z_j), (x_k, y_i, z_{j+1}), (x_{k+1}, y_{i+1}, z_j), (x_{k+1}, y_i, z_{j+1}), (x_k, y_{i+1}, z_{j+1}), $(x_{k+1}, y_{i+1}, z_{j+1})$. This condition takes the following form:

```
V1 = (F1_Nodes[i][j] XOR F1_Nodes[i][j]<<1) OR
(F1_Nodes[i][j] XOR F1_Nodes[i+1][j]) OR
(F1_Nodes[i][j] XOR F1_Nodes[i][j+1]) OR (F1_
Nodes[i][j] XOR F1_Nodes[i+1][j<<1) OR
(F1_Nodes[i][j] XOR F1_Nodes[i][j+1]<<1) OR (F1_
Nodes[i][j] XOR F1_Nodes[i+1][j+1])
(F1_Nodes[i][j] XOR F1_Nodes[i+1][j+1]<<1).   (8)
```

If among eight tops there are those where the function takes on different signs, then the result of this expression will be 1, otherwise – 0. The result of comparison (NOT (F1_Nodes[i][j])) AND F1_NegExtremums[i][j] will be 1 of only the zero bits in the element of array F1_Nodes[i][j], that is, of those from L described by this element tops, in which the function has an integral sign (and, accordingly, the change in signs is found). The result of comparison (F1_Nodes[i][j] AND F1_PosExtremums[i][j]) will be 1 only for those bits F1_Nodes[i][j] that represent the tops where the function takes negative values, while unity in the corresponding bits of array F1_PosExtremums[i][j] means a negative sign of the function. A general condition of change in the sign of function in the tops or critical points will take the form:

```
V1 = V1 OR ((NOT(F1_Nodes[i][j]))
AND F1_NegExtremums[i][j]) OR
(F1_Nodes[i][j]) AND F1_PosExtremums[i][j]).   (9)
```

It is important that every bit in array F1_Nodes sets only one top of parallelepiped, not all eight, since if among eight tops there are tops with a different sign, value of the expression will be 1; otherwise – all tops have the same sign and it does not matter, the sign of which of them we compare.

Similarly, we form values V2 and V3. Parallelepiped can contain points of quadrics intersection (1), if

```
V1 AND V2 AND V3                              (10)
```

is valid.

4. 3. Algorithm for the localization of the intersection points of three quadrics

1. Define closed region D, in which we will look for the solutions. In a general case, the base for selection can be the knowledge about the measurement range. In this case, D will have the shape of cube because each coordinate of the vector quantity can intersect any values from zero to the module of searched vector quantity (but, without loss of generality, we will deal with a rectangular parallelepiped). For example, if module of the measured quantity may be in the range from V_{min} to V_{max}, then we will search for each of the coordinates in the region from $-V_{min}$ to V_{max}. If, among the preset quadrics, there are limited surfaces (e. g., ellipsoids), region D can be determined by determining to which of 17 types

of quadrics each of the quadrics (1) belongs, and reducing quadric equation to canonical form.

2. Set L, M and N points, by which the Ox, Oy and Oz coordinate axes are split, respectively. For convenience of work with bits and arrays of elements of the scalar types, each of these numbers should not exceed the number of bits in the unsigned integer type of data (for reasons outlined above).

3. Create arrays F1_Nodes, F2_Nodes, F3_Nodes, F1_PosExtremums, F1_NegExtremums, F2_PosExtremums, F2_NegExtremums, F3_PosExtremums, F3_NegExtremums and RootsLocalized and initialize them with zeros.

4. For each combination of values x_k, y_i and z_j ($k=\overline{1,L}$, $i=\overline{1,M}$, $j=\overline{1,N}$), using the three cycles, it is necessary to:

4. 1. find value of each of functions (2) and assign bit with number k with indices i and j to the appropriate array (F1_Nodes, F2_Nodes or F3_Nodes) if the function is negative in the point, otherwise – 0;

4. 2. calculate by formulas (7) such values of x, y and z for each of three functions so that in points (x,y_i,z_j), (x_k,y,z_j) and (x_k,y_i,z) equations (4) are converted to equalities, as well as to compute the signs of functions (2) in these points;

4. 3. find such values k_1, i_1, j_1 so that $x\in[x_{k1},x_{k1+1}]$, $y\in[y_{i1},y_{i1+1}]$ and $z\in[z_{j1},z_{j1+1}]$, that is, determine to which edges of parallelepipeds the points of extremum, calculated in step 4. 2, belong;

4. 4. set to 1 bit with number k_1 of the element with indices i and j, bit with number k of the element with indices i_1 and j and bit with number k of the element with indices i and j_1 in the appropriate array (one of F1_PosExtremums, F1_NegExtremums, F2_PosExtremums, F2_NegExtremums, F3_PosExtremums and F3_NegExtremums, depending on the analyzed function and its sign);

4. 5. set the bits for all the "neighbors" (parallelepipeds, which also contain the found edges); there may be 9, 3 or 0, in accordance with the above stated considerations;

4. 6. find such combinations of values y_1 and z_1, x_1 and z_2, x_2 and y_2 for each function so that in points (x_k,y_1,z_1), (x_1,y_i,z_2) and (x_2,y_2,z_j) equations (5) are converted to equalities;

4. 7. find such values i_1 and j_1, k_1 and j_2 i k_2 and i_2 so that $y_1\in[y_{i1},y_{i1+1}]$ and $z_1\in[z_{j1},z_{j1+1}]$, $x_1\in[x_{k1},x_{k1+1}]$ and $z_2\in[z_{j2},z_{j2+1}]$, $x_2\in[x_{k2},x_{k2+1}]$ and $y_2\in[y_{i2},y_{i2+1}]$. Find the signs of functions (2) in the points found in step 4. 6 and set unities in the corresponding bits of arrays to track the signs of functions in the points of extremum by the technique, similar to the one described in step 4. 4;

4. 8. set the bits for parallelepipeds with the sides defined in step 4. 7 (there may be 3 or 0 of such parallelepipeds).

5. Form element RootsLocalized[i][j] using (8)–(10) for each combination of indices i and j.

6. Compute critical points of functions (2), identify intervals to which they belong, and complement generated array RootsLocalized with data on the signs of functions (2) in critical points.

7. The algorithm is finalized.

5. Results of comparative analysis of the proposed algorithm to the algorithm based on interval arithmetic

To determine the appropriateness of the proposed algorithm, we conducted a series of numerical experiments of implementations of two algorithms – the one proposed in this work and an algorithm based on interval arithmetic [14].

The algorithm was implemented in the language C in the Keil uVision 5 environment; its verification was performed at the STM32F407VG microcontroller (as part of the evaluation board STM32F4Discovery), built on the base of the ARM Cortex-M4 architecture. The algorithm based on interval arithmetic [14] is realized in the same language using the same compiler and the same microcontroller, which means a possibility of correct comparison of these two algorithms.

A comparison of implementations of the algorithms was carried out by the following attributes: share of the regions that were found as a result of work of the algorithm but they did not actually contain the points of quadrics intersection, time of execution, volume of machine code and the use of operative memory.

As the test sets, we took 48 systems of three quadric equations whose points of intersection are known. Manual testing revealed that none of the solutions of the systems of equations was "overlooked" – for each of the test systems of quadric equations among the results of the algorithm's work there were regions (rectangular parallelepipeds), which contained the points of quadrics intersection. However, similar to the case with the algorithm described in [14], some of the found regions happened to be a "false alarm" – in fact, they did not contain solutions for the test systems of equations.

Initially we tested the simplest variants for the systems of equations, for which it was easy to find analytical solutions – systems of canonical equations of quadrics (without non-linear carry and rotation of the axes). In such equations, part of coefficients in the variables, their squares and pairwise products are equal to zero, which allowed us to test the branches of code associated with dividing by zero. A comparison of the regions found in the course of work of the two algorithms was performed at the similar test sets with the same node points (otherwise, results of the comparison would not be considered valid). Results are grouped in Table 1.

Next, rotation of axes and linear carry were applied to the same systems of equations.

Since each quadric equation in the system underwent the same conversion, the number of intersection points remained – only their coordinates in the "old" coordinate system changed. For each of 12 test systems of equations represented in Table 1, the rotation was performed at three different combinations of angles. Test results appeared to be about the same as in Table 1, with possible differences not exceeding a few unities.

Asymptotic estimation of complexity of both algorithms in the notation "O large" is the same, but the nature of instructions is different, that is, the set of machine instructions is substantially different, thus the actual execution time may vary significantly.

An evaluation of execution time of implementations of two algorithms at the same number of node points (that is, at the identical sets of numbers L, M and N) was performed by practical way only – using measurements of time of realization of each of the algorithms (at the same clock frequency) at the same test sets. The algorithm proposed in this paper proved to be slower on average by 12 %.

Table 1

Numbers of regions of the points of quadrics intersection, localized by the implementations of two algorithms

No.	System of quadrics equations	Solutions	Quantity of regions found using the algorithm of interval arithmetic		Quantity of regions found using the proposed algorithm	
			L = 8 M = 8 N = 8 (343 regions)	L = 16 M = 16 N = 16 (3375 regions)	L = 8 M = 8 N = 8 (343 regions)	L = 16 M = 16 N = 16 (3375 regions)
1	$\begin{cases} x^2+y^2-1=0 \\ x^2+z^2-1=0 \\ x^2+y^2+z^2-1=0 \end{cases}$	$(-1,0,0),\ (1,0,0)$	42	56	8	26
2	$\begin{cases} x^2+y^2+z^2-2=0 \\ x^2+0.05y^2+z^2-2=0 \\ -x^2+y^2+2z^2-4=0 \end{cases}$	$(0,0,-\sqrt{2}),\ (0,0,\sqrt{2})$	40	58	24	32
3	$\begin{cases} x^2+y^2+z^2-2=0 \\ x^2+0.05y^2+z^2-2=0 \\ -x^2+y^2+2z^2+4=0 \end{cases}$	$(-\sqrt{2},0,0),\ (\sqrt{2},0,0)$	40	58	24	32
4	$\begin{cases} x^2+y^2+z^2-2=0 \\ x^2+0.05y^2+z^2-2=0 \\ x^2-2y^2+z^2-2=0 \end{cases}$	a set of all points $(x, 0, z)$, so that $x^2+y^2=2$	142	182	64	64
5	$\begin{cases} x^2+y^2+z^2-2=0 \\ x^2+0.05y^2+z^2-2=0 \\ x^2+z^2+4x+3=0 \end{cases}$	$\left(-\frac{5}{4},0,-\frac{\sqrt{7}}{4}\right),\ \left(-\frac{5}{4},0,\frac{\sqrt{7}}{4}\right)$	36	52	24	32
6	$\begin{cases} x^2+0.05y^2+z^2-2=0 \\ x^2+y^2+z^2-2=0 \\ x^2+z^2+y^2-1=0 \end{cases}$	no intersection points	28	46	8	8
7	$\begin{cases} x^2+0.05y^2+z^2-4=0 \\ x^2+y^2+z^2-4=0 \\ x^2+y^2+z^2-2x=0 \end{cases}$	$(2,0,0)$	42	50	24	32
8	$\begin{cases} x^2+0.05y^2+z^2-4=0 \\ x^2+y^2+z^2-4=0 \\ x^2+y^2+z^2+2x=0 \end{cases}$	$(-2,0,0)$	42	50	24	32
9	$\begin{cases} x^2+y^2-2z=0 \\ x^2+y^2+z^2-2z-4=0 \\ x^2+z^2-4z+3=0 \end{cases}$	$(-1,-\sqrt{3},2),\ (-1,\sqrt{3},2),$ $(1,-\sqrt{3},2),\ (1,\sqrt{3},2)$	46	54	24	24
10	$\begin{cases} x^2+y^2-2z=0 \\ x^2+y^2+z^2-2z-4=0 \\ y^2+z^2-4z+4=0 \end{cases}$	$(-2,0,2),\ (2,0,2)$	44	54	24	24
11	$\begin{cases} x^2+y^2-2z=0 \\ x^2+y^2+z^2-2z-4=0 \\ y^2+z^2-4z+2=0 \end{cases}$	$(-\sqrt{2},-\sqrt{2},2),\ (\sqrt{2},-\sqrt{2},2),$ $(-\sqrt{2},\sqrt{2},2),\ (\sqrt{2},\sqrt{2},2)$	46	58	24	24
12	$\begin{cases} x^2+y^2+z^2-4=0 \\ x^2-y^2+z^2=0 \\ x^2+y^2-z^2=0 \end{cases}$	$(0,-\sqrt{2},-\sqrt{2}),\ (0,-\sqrt{2},\sqrt{2}),$ $(0,\sqrt{2},-\sqrt{2}),\ (0,\sqrt{2},\sqrt{2})$	46	58	16	16

The volume of machine code and engaged operative memory was estimated by data taken from map-file generated by the Keil uVision environment after compiling and linking. The amount of permanent memory, used by the algorithm proposed in this work, appeared to be 20 % larger than the analogous indicator for the algorithm based on interval arithmetic. Instead, the volume of the engaged operative memory was 15 % less.

Appropriateness of the use of the algorithm based on interval arithmetic as an intermediate link in the process of solving the systems of polynomial equations has already been proven in [14]. Test sets for verification of the algorithm proposed in this work included the majority of the systems of quadrics equations, by which the algorithm in [14] was tested, and the test was carried out particularly in the same region of the search for solutions and in the same node points. For this reason, the role of the proposed algorithm in refining the solutions was not examined in this work.

6. Discussion of results of comparative analysis of the proposed algorithm to the algorithm based on interval arithmetic

Results of testing implementations of the two compared algorithms allow us to state that the new algorithm achieved a reduction in the quantity of irrelevant regions with potential points of intersection of three quadrics by approximately 2.2 times. It should be noted, however, that the number of regions depends on ranges $[x_a, x_b]$, $[y_a, y_b]$, $[z_a, z_b]$, in this case, the narrowing of ranges does not always lead to an increase (or, conversely, decrease) in the number of regions. In addition, the number of "parasite" regions appeared to be different for different test sets. A dependence of the "parasite" regions on the system of quadrics equations is explained by the principle of the algorithm's work – it is guaranteed to "reject" only those regions in which at least one pair of quadrics has no intersection points. That is, all regions containing the points of intersection of each pair of quadrics definitely belong to the list of regions that could potentially contain the sought solutions for the systems of quadrics equations. However, it is quite possible that in a certain region all quadrics intersect in pairs, but not in one common point. The fact that the algorithm based on interval arithmetic yields a larger percentage of "parasite" regions is consistent with the results of applying interval arithmetic to the estimation of errors of measurement – they are always overstated and indicate the "worst case". Behavior of the algorithms was studied for the test cases with one, two, four and multitude of intersection points of three quadrics, as well as in the absence of intersection points. Limitation of the test sets allows us to speak only about approximate percentages of winning in the relevance of results of the proposed algorithm relative to the algorithm based on interval arithmetic.

However, conducted numerical experiments allow us to affirm that in case of necessity to solve a system of quadrics equations, the proposed algorithm is more expedient than its analogs for the implementation in micro programming software, taking into account constraints of their resources. Obtained results are focused primarily on the application in intelligent sensors of vector quantities; however, they can be extended to other problems that comprise equations of surfaces of the second order and their systems. Moreover, the same technique can be applied to any of the functions of three variables, the finding of partial derivatives for which has the same complexity as for the functions that represent the left parts of equations (1). If the computation of partial derivatives requires more resources (for example, in the case of equations of the highest degree) or there is no an analytical method for finding critical points of functions that represent the left parts of the solved system of equations, then it requires additional study.

As noted above, a new algorithm requires more of permanent memory and runs longer. Presented numerical indices are not absolute since they apply only to a single model of microcontroller. Generalized results may be obtained theoretically if we consider the number of instructions in the algorithm and the period of implementation of each of the instructions for a particular architecture of microcontroller.

Further direction of research is the reduction in computational complexity of the proposed algorithm. A potential way to achieve this result is a breakdown of the region into intervals of different step, in contrast to the division into intervals of equal length.

7. Conclusions

1. We performed an analysis and systematization of mathematical apparatus for the localization of intersection points of quadrics, which allowed us to design a new algorithm for the localization of quadrics intersection points that is based on the properties of continuous differentiable functions in closed regions and verification of the signs of functions in the node and critical points. The developed algorithm was implemented for a microcontroller of the ARM Cortex-M4 architecture.

2. Conducted comparative analysis of the new algorithm to an algorithm-analogue allows us to assert that the new algorithm provides for higher relevance results, approximately by 2.2 times, of the found regions with potential points of intersection at the execution longer by 12 %. The new algorithm requires 20 % more of permanent memory, but 15 % less of operative memory. Thus, the studies we conducted have proved the expediency of applying the proposed algorithm.

References

1. Bol'shakova, I. A. Mikroelektronni sensorni prystroi mahnitnoho polia [Text] / I. A. Bol'shakova, M. R. Gladun, R. L. Goljaka, Z. Ju. Gotra, I. Je. Lopatyns'kyj, Je. Potencki, L. I. Sopil'nyk; Z. Ju. Gotra (Ed.). – Lviv: Vyd. Natsionalnoho universytetu «Lvivska politekhnika», 2001. – 412 p.
2. Bolshakova, I. A. Metody modeliuvannia ta kalibruvannia 3D-zondiv mahnitnoho polia na rozshcheplenykh khollivskykh strukturakh [Text] / I. A. Bolshakova, R. L. Holyaka, Z. Y. Hotra, T. A. Marusenkova // Elektronika ta zviazok. Tematychnyi vypusk «Elektronika ta nanotekhnolohii». – 2011. – Issue 2 (61). – P. 34–38.

3. Riviere, J. M. Design of smart sensors: towards an integration of design tools [Text] / J.-M. Riviere, D. Luttenbacher, M. Robert, J.-P. Jouannet // Sensors and Actuators A: Physical. – 1995. – Vol. 47, Issue 1-3. – P. 509–515. doi: 10.1016/0924-4247(94)00952-e
4. Bowen, M. Consideration for the design of smart sensors [Text] / M. Bowen, G. Smith // Sensors and Actuators A: Physical. – 1995. – Vol. 47, Issue 1-3. – P. 516–520. doi: 10.1016/0924-4247(94)00953-f
5. Chaudhari, M. Study of Smart Sensors and their Applications [Text] / M. Chaudhari, S. Dharavath // International Journal of Advanced Research in Computer and Communication Engineering. – 2014. – Vol. 3, Issue 1. – P. 5031–5034.
6. Buchberger, B. Bruno Buchberger's PhD thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal [Text] / B. Buchberger // Journal of Symbolic Computation. – 2006. – Vol. 41, Issue 3-4. – P. 475–511. doi: 10.1016/j.jsc.2005.09.007
7. Hashemi, A. Applying IsRewritten criterion on Buchberger algorithm [Text] / A. Hashemi, B. M.-Alizadeh // Theoretical Computer Science. – 2011. – Vol. 412, Issue 35. – P. 4592–4603. doi: 10.1016/j.tcs.2011.04.040
8. Eder, C. F5C: a variant of Faugere's F5 algorithm with reduced Grobner bases [Text] / C. Eder, J. Perry // Journal of Symbolic Computation. – 2010. – Vol. 45, Issue 12. – P. 1442–1458. doi: 10.1016/j.jsc.2010.06.019
9. Gao, S. A new incremental algorithm for computing Groebner bases [Text] / S. Gao, Y. Guan, F. Volny // Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation – ISSAC '10. – 2010. doi: 10.1145/1837934.1837944
10. Hashemi, A. Invariant G2V algorithm for computing SAGBI-Grobner bases [Text] / A. Hashemi, B. M.-Alizadeh, M. Riahi // Science China Mathematics. – 2012. – Vol. 56, Issue 9. – P. 1781–1794. doi: 10.1007/s11425-012-4506-8
11. Wilson, D. J. Speeding up cylindrical algebraic decomposition by Gröbner bases [Text] / D. J. Wilson, R. J. Bradford, J. H. Davenport // Lecture Notes in Computer Science. – 2012. – P. 280–294. doi: 10.1007/978-3-642-31374-5_19
12. Dupont, L. Near-optimal parameterization of the intersection of quadrics: III. Parameterizing singular intersections [Text] / L. Dupont, D. Lazard, S. Lazard, S. Petitjean // Journal of Symbolic Computation. – 2008. – Vol. 43, Issue 3. – P. 216–232. doi: 10.1016/j.jsc.2007.10.007
13. Dupont, L. Complete, Exact and Efficient Implementation for Computing the Adjacency Graph of an Arrangement of Quadrics [Text] / L. Dupont, M. Hemmer, S. Petitjean, E. Schomer // Lecture Notes in Computer Science. – 2007. – P. 633–644. doi: 10.1007/978-3-540-75520-3_56
14. Marusenkova, T. A. Development of algorithm and embedded software for separation of intersection points of quadrics [Text] / T. A. Marusenkova, D. O. Horman // Eastern-European Journal of Enterprise Technologies. – 2015. – Vol. 3, Issue 4 (75). – P. 16–20. doi: 10.15587/1729-4061.2015.42609
15. Marusenkova, T. Approach To Roots Separation For Solving Nonlinear Equations On ARM Cortex-Based Microcontrollers [Text] / T. Marusenkova, I. Yurchak // XXII Ukrainian-Polish Conference on "CAD in Machinery Design. Implementation and Educational Issues". – Lviv, 2014. – P. 101–103.
16. Horman, D. Algorithm of Polynomials' Root Separation for ARM-Based Microcontrollers [Text] / D. Horman, T. Marusenkova, I. Yurchak // XIII International Conference The Experience of Designing and Application of CAD Systems in Microelectronics. – Lviv-Poljana, 2015. – P. 50–52.