UDC 004.4, 519.688

T. Marusenkova, I. Yurchak
Lviv Polytechnic National University

AN ALGORITHM CHECKING THE UNIQUENESS OF QUADRICS' INTERSECTION POINT FOR VECTOR QUANTITIES SMART SENSORS

The paper considers vector quantities sensors possessing output characteristics described by quadrics' equations. The vector coordinates can be found as the quadrics' intersection points. Measurement results are trustworthy if there is one intersection point. We propose an algorithm detecting whether a set of three quadrics' equations has the only solution. Its implementation is intended for vector quantity smart sensors' firmware.

Keywords: algorithm, smart sensors, microcontroller, root separation, quadrics' intersection points, vector quantity, number of solutions.

АЛГОРИТМ ВИЗНАЧЕННЯ ЄДИНОСТІ ТОЧКИ ПЕРЕТИНУ КВАДРИК ДЛЯ ІНТЕЛЕКТУАЛЬНИХ СЕНСОРІВ ВЕКТОРНИХ ВЕЛИЧИН

Розглянуто проблематику використання сенсорів векторних величин, польові характеристики яких є функціями, що описують квадрики. Координати вектора вимірюваної величини визначаються як точки перетину квадрик. Таких точок може бути декілька, що ставить під сумнів придатність результатів вимірювання. Запропоновано алгоритм визначення єдиності розв'язку системи трьох рівнянь квадрик. Реалізацію алгоритму можна використати у мікропрограмному забезпеченні інтелектуальних сенсорів векторних величин.

Ключові слова: алгоритм, інтелектуальні сенсори, мікроконтролер, локалізація, точки перетину квадрик, векторна величина, кількість розв'язків.

Problem statement

Vector quantities sensors are an essential part of data acquisition for further processing this data. New design solutions of such sensors have been constantly developed and investigated in order to improve sensors' sensibility (and thus measurement accuracy) and reducing sensors' size (and thus increasing the density of mapping the field being measured) [1].

However, improvement of any sensors' properties scarcely can be achieved without impairment of other properties. Particularly, output characteristics of the elementary actuators comprising such a new-generation sensor can depend on all the three coordinates of the vector being measured, their squares and their pairwise products. As the result, there comes up a problem of fetching the sought vector coordinates from the sensor's output characteristics, causing the necessity to solve a set of non-linear equations [2]. Moreover, the output characteristics' coefficients may differ from one sensor specimen to another. This fact complicates the calibration procedure of such sensors and their usage in practice, since in general case a set of non-linear equations can have multiple solutions. At best only one of these solutions coincides with the "actual" coordinates of the vector being measured. In order to find out which solution is suitable, some additional information is needed. It can be some auxiliary dependencies or information about the nature of the sensor and the quantity being measured.

In order to reduce complexity of measurement results, a sensor can be fitted with a "brain" which is typically a microcontroller. Thus sensors become "smart" [3–5]. Firmware of smart sensors, as of any

embedded system, should meet more strict requirements than general-purpose computers' software. These requirements cover execution time (and thus computational complexity), consumed RAM and ROM and reliability. An important part of processing the measured data is development of algorithms and firmware for detection whether the set of equation defined by the output characteristics of a vector quantity sensor has the only solution or not. If not measurement results should be discarded.

Recent research and publications analysis

The work [6] introduces a method for evaluating the applicability of the magnetic field measurement results if they are obtained using a 3D-probe comprised of three sensors with output characteristics that depend on all the components of a magnetic field and their squares. The work proposes simple analytic dependencies that allow us to detect whether it's possible to measure a magnetic field with inaccuracy not exceeding some defined value using a 3D-probe with known output characteristics' coefficients. The same analytic dependencies are applicable for evaluating measurement results obtained by other vector quantities' sensors if their output characteristics are described by the same function as the 3D-probe's that had been investigated. The physical meaning of the output characteristics' coefficients may be left out of the scope. However, the proposed analytic dependencies do not make allowances for the vector coordinates' pairwise products. Besides, they were deduced under a range of assumptions, all of them related to "the worst possible case". This means that if some measurement results fit these analytic dependencies, they can be considered applicable, but otherwise it's too early to draw any conclusions about the results' applicability – additional investigations are required.

In cases when analytic solutions cannot be obtained the sought vector's coordinates are to be found as a solution for the set of non-linear equations that describe the output characteristics of a sensor using numerical methods.

In [7] it was shown how roots separation influence the results of using library functions implementing well-known numerical methods for finding exact solutions of sets of non-linear algebraic equations. These investigations were carried out on ARM based microcontrollers. All the library functions refine some approximate solutions that should be passed to the functions as their parameters. These solutions can be given as intervals or point values. It was proved that library implementations of numerical methods can fail to find all the solutions for an equation set or even find faulty solutions if they were given too roughly formed approximate solutions. An algorithm of roots separation for a set of quadrics' equations was proposed in [7]. The algorithm has complexity $O\left(n^3\right)$ and ensures that all the regions that potentially contain the equation set's solutions would be found. The algorithm's disadvantage is that some of the regions found as the result of its work actually do not contain any solutions for an equation set. In practice, along with the problem of finding all the solutions for an equation set there exists a problem of finding out how many solutions an equation set has. Particularly, as it was mentioned earlier, if an equation set characterizing a sensor has more than one solution, the sensor cannot be used for measurements without any additional data, since otherwise there are no means to know, which of the found solutions is "correct".

In order to find out the amount of solutions for non-linear equations and equation sets the modern mathematics applies Buchberger's algorithm that finds Gröbner basis of the ideal in a finite number of steps [8–9]. The advantage of Buchberger's algorithm is its universality – it's suitable for any nonlinear algebraic equation or an equation set. However, Buchberger's algorithm assumes symbolical calculations. There exist rather accurate implementations of the algorithm by numerical methods. However, they are susceptible to how accurate the results of interim calculations are saved. Hence, in general the idea of implementing Buchberger's algorithm in firmware contradicts with one of the main concepts of programming embedded systems – the less resource firmware consumes, the better. Moreover, the results of all the efforts related to implementation of Buchberger's algorithm turn to be redundant in case if all that we need to know is whether an equation set has the only solution or not.

Statement of purpose

This work is aimed at development of a simple, "quick" algorithm that detects whether a set of quadrics' equations has the only solution and is appropriate for implementation as a part of firmware for

ARM based microcontrollers, taking into account restrictions of programming embedded systems. The algorithm and firmware based on it are intended for making a conclusion upon the measurement results obtained from vector quantity sensors with output characteristics described by quadrics' equations.

A method for detecting quadrics' intersection points

Let's suppose that a smart sensor of some vector quantity contains three actuators, output characteristics of which can be described by quadrics' equations:

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

here x, y and z are projections of the vector quantity being measured onto axes Ox, Oy and Oz correspondingly of some Cartesian coordinate system assigned with the package of the sensor; a_{ij}, b_{ij}, c_{ij} ($i=\overline{1,3}$, $j=\overline{1,3}$) are known coefficients of the output characteristics of three actuators comprising the sensor; S_i are the signals read from the actuators.

We'll look for solutions of equation set (1) in some region R having the shape of a rectangular parallelepiped comprised by all the points $x\in[x_a,x_b]$, $y\in[y_a,y_b]$, $z\in[z_a,z_b]$. The region's selection can be caused by some available knowledge about the measurement range. In this case R is cube, since each coordinate of the sought vector can have any value from the negative value of the vector's length to its positive value. One of approaches to define region R is to determine which type of 17 possible quadric types each of our quadrics (1) belongs to, represent them in their canonical form and find out their dimensions. Let's take L, M and N equidistant points in ranges $[x_a,x_b]$, $[y_a,y_b]$ and $[z_a,z_b]$ correspondingly. The first and the last points on each axis will be lower and upper boundaries of the intervals. Thus the whole region R can be represented as a set of rectangular parallelepipeds $x\in[x_k,x_{k+1}]$, $y\in[y_i,y_{i+1}]$, $z\in[z_j,z_{j+1}]$ ($k=\overline{1,L-1}$, $i=\overline{1,M-1}$, $j=\overline{1,N-1}$) with vertices coinciding with the nodes obtained by above mentioned division of ranges by equidistant points.

Let's consider the first equation in equation set (1) at two fixed points $y=y_0$ and $z=z_0$:

$$a_{11}x^2 + a_{12}y_0^2 + a_{13}z_0^2 + b_{11}xy_0 + b_{12}xz_0 + b_{13}y_0z_0 + c_{11}x + c_{12}y_0 + c_{13}z_0 - S_1 = 0 \quad (2)$$

Equation (2) is a quadratic equation where x is unknown, thus it can have 0, 1 or 2 real roots.

Let f denote the function that represents the left side of equation (2) at points x_k ($k=\overline{1,L-1}$) and f_0 denote the value of the function at x_0. Hence the value of function f at some point x_K ($x_K=x_0+\Delta x\cdot K$), where Δx is the distance between the equidistant points in range $[x_a,x_b]$, can be expressed as follows:

$$\begin{aligned} f_N = f(x_0+K\Delta x) = a_{11}x_0^2 + 2Ka_{11}\Delta x x_0 + K^2a_{11}(\Delta x)^2 + a_{12}y_0^2 + a_{13}z_0^2 + b_{11}x_0y_0 + Kb_{11}\Delta x y_0 + \\ + b_{12}x_0z_0 + Kb_{12}\Delta x z_0 + b_{13}y_0z_0 + c_{11}x_0 + Kc_{11}\Delta x + c_{12}y_0 + c_{13}z_0 - S_1 = \\ = f_0 + 2Ka_{11}\Delta x x_0 + K^2a_{11}(\Delta x)^2 + Kb_{11}\Delta x y_0 + Kb_{12}\Delta x z_0 + Kc_{11}\Delta x \end{aligned}$$

We are looking for such a number K that $f(x_K)=0$, solving the quadratic equation:

$$a_{11}(\Delta x)^2K^2 + (2a_{11}\Delta x x_0 + b_{11}\Delta x y_0 + b_{12}\Delta x z_0 + c_{11}\Delta x)K + f_0 = 0 \quad (3)$$

The solution is expressed as:

$$K_{1,2} = \frac{-(2a_{11}\Delta x x_0 + b_{11}\Delta x y_0 + b_{12}\Delta x z_0 + c_{11}\Delta x) \pm \sqrt{D}}{2a_{11}(\Delta x)^2} \quad (4)$$

where D is the discriminant of equation (3). Similar formulas can be obtained for the cases when y or z are free variables and the couples of variables x and z or x and y are fixed correspondingly. The same approach relates to the second and the third equations as well.

Since K is the number of a range that contains such a point x, that it is a solution for equation (3), we'll round the found numbers to the nearest less natural numbers. Negative values and values exceeding the amount of ranges L should be discarded for they do not belong to the region R where we try to find the equation set's solutions.

Since for fixed values of any couple of variables (x and y, x and z or y and z) the value of the remaining third variable that turns equation (2) into equality, can be found from dependencies alike (4), instead of considering combinations of x_k, y_i and z_j ($k=\overline{1,L}$, $i=\overline{1,M}$, $j=\overline{1,N}$) it's enough to examine combinations of each couple of variables (x and y, x and z or y and z). Thus in contrast to the algorithm introduced in [7] using an approach with solving quadratic equations we can reduce the computational complexity from $O\left(n^3\right)$ to $O\left(n^2\right)$. This reduction would allow us to increase numbers L, M and N in comparison with the values of L, M and N "affordable" at the computational complexity $O\left(n^3\right)$. This leads to reduction of distances Δx, Δy and Δz between equidistant points on any of the coordinate axes. If ΔV is the allowed inaccuracy of measuring the length of the sought vector V and values Δx, Δy and Δz are small enough to fit this inaccuracy, then the found intervals $[x_k, x_{k+1}]$, $[y_i, y_{i+1}]$, $[z_j, z_{j+1}]$ ($k=\overline{1,L-1}$, $i=\overline{1,M-1}$, $j=\overline{1,N-1}$) can be roughly considered the solution for equation set (1). In this case there is no need in refinement of this solution by numerical methods implemented in library functions suitable for firmware.

An algorithm for check if quadrics have the only intersection point

Step 1. To define some region R where we'll seek for quadrics' intersection points. This region will be a rectangular parallelepiped comprised by all the points $x \in [x_a, x_b]$, $y \in [y_a, y_b]$, $z \in [z_a, z_b]$.

Step 2. To set values L, M and N for division of the intervals $[x_a, x_b]$, $[y_a, y_b]$ and $[z_a, z_b]$ by L, M and N equidistant points correspondingly. To evaluate the distances Δx, Δy and Δz between the neighbor points on three axes.

Step 3. To create an array F<i>_Zeros[M–1][N –1] of unsigned integer items for each equation in equation set (1). Here we'll keep information about regions containing intersection points of the quadrics (1). The first index of each array is the index of an interval on coordinate axis Oy, the second index is the index of an interval on coordinate axis Oz. The letter "i" in angular brackets stands for the index of an equation in equation set (1). Thus, we'll have three arrays – F1_Zeros, F2_Zeros and F3_Zeros. At first all the items of each array should be initialized by some constant number X_NA, which indicates that there are no solutions for the corresponding equation of equation set (1) in the region represented by any specific array item. If during algorithm's execution this assumption about the absence of the solutions is denied, X_NA will be replaced by the index of an interval on coordinate axis Ox where a solution was found. Otherwise the value of an array's item will still remain equal to X_NA. Since there can exist 0, 1 or 2 possible interval indices (due to the fact that there are 0, 1 or 2 possible solutions for any quadratic equation), we can use the high-order byte and the low-order byte of any array's item to store different indices (under the assumption that the arrays type uses 16 bits and we do not need more than $L=256$). More generally, we can use "high half a type width" and "low half a type width" for storing indices. For $L=256$ we can use uint16_t. An alternative is to split data to six, not three arrays, however it's unlikely that more than 256 equidistant points are needed due to the fact that the algorithm slows down in this case.

Step 4. For each combination of i and j, i.e. for y_i and z_j ($i=\overline{1,M}$, $j=\overline{1,N}$), using formulas (4) we calculate such indices k_l ($k_l=\overline{1,L-1}$) of intervals that point $\left(x_a + k_l \cdot \Delta x, y_i, z_j\right)$ is a solution for the first equation in equation set (1). If these intervals are out of the range from 1 to $L-1$, we should discard them. Otherwise we store them in the high-order byte and the low-order byte of item F1_Zeros[i][j]. Similarly, we calculate and store the indices of the intervals containing the solutions for the second and the third equations in equation set (1), in items F2_Zeros[i][j] and F3_Zeros[i][j].

Step 5. For each combination of values k and i, i.e., for x_k and y_i ($k = \overline{1, L}$, $i = \overline{1, M}$), using formulas alike (4) we calculate such indices j_l ($j_l = \overline{1, N-1}$) of the intervals on coordinate axis Oz, that point $(x_k, y_i, z_a + j_l \cdot \Delta z)$ is a solution for the first equation in equation set (1). Then we check whether the found numbers are within the allowed range. If not, they should be discarded. Otherwise each of the found values becomes the second index of an array item with the first index i: F1_Zeros[i][j_l]. The low-order byte of this array item if it's equal to X_NA should be assigned to k (the index of the considered interval on coordinate axis Ox). If the low-order byte contains a number different from X_NA, it means that during some previous iteration such a combination of interval indices k, i and j has been found that the rectangular parallelepiped defined by this indices combination contains a solution for the first equation of equation set (1). In this case we should write number k into the high-order byte of F1_Zeros[i][j_l] and not in the low-order byte. Similarly, we should find the indices of intervals on axis Oz that form solutions for the second and the third equations in equation set (1) and store the found values in F2_Zeros and F3_Zeros.

Step 6. For each combination of values k and j, i.e., for x_k and z_j ($k = \overline{1, L}$, $j = \overline{1, N}$), using formulas alike (4) we calculate such indices i_l ($i_l = \overline{1, M-1}$), that point $(x_k, y_a + i_l \cdot \Delta y, z_j)$ is a solution for the first equation in equation set (1). The found values should be the first indices of array items with the second index j, if they are inside their correct range. For forming items of F1_Nodes we apply the same approach as in step 5. In the same way, we should fill arrays F2_Nodes and F3_Nodes with data on regions containing solutions for the second and the third equations in equation set (1). As the result of the first six steps of the algorithm for those regions containing solutions for each separate equation in equation set (1) the items of three arrays, F1_Nodes, F2_Nodes and F3_Nodes, representing these regions will be populated by values different from X_NA.

Step 7. For each combination of values i and j ($i = \overline{1, M-1}$, $j = \overline{1, N-1}$), we should compare items F1_Nodes[i][j], F2_Nodes[i][j] and F3_Nodes[i][j]. If the content l of the high-order byte or the low-order-byte in F1_Nodes[i][j] coincides with the content of the high-order byte or the low-order byte of F2_Nodes[i][j] and F3_Nodes[i][j], then the rectangular parallelepiped comprised of all the points $x \in [x_l, x_{l+1}]$, $y \in [y_i, y_{i+1}]$, $z \in [z_j, x_{j+1}]$ ($l = \overline{1, L-1}$, $i = \overline{1, M-1}$, $j = \overline{1, N-1}$) potentially contains solutions for equation set (1). However, it's possible that inside a parallelepiped there are pairwise quadrics' intersection points but still there is no a single point where all the quadrics intersect. Nevertheless, we increase the counter for equation set's solutions.

Step 8. If the counter for equation set's solutions is greater than 1 and values Δx, Δy and Δz are small enough to fit the allowed inaccuracy of the vector quantity measurement, we can consider the found rectangular parallelepipeds to be approximate solutions for equation set (1) and conclude that the obtained measurement results are not applicable.

Step 9. If the counter for equation set's solutions is greater than 1 and values Δx, Δy and Δz do not fit the allowed inaccuracy of the vector quantity measurement, we can refine the found approximate solutions (i.e. found parallelepipeds that potentially contain quadrics' intersection points) by available library functions suitable for usage in firmware. The refined solutions are to be substituted into equation set (1) to check whether they are actual solutions for the equation set. If for some pair of the refined solutions the left side of equations (1) differs from their right side by a value within the allowed inaccuracy (nearly equal to zero), we can conclude that the obtained measurement results are not applicable.

Step 10. If the counter for equation set's solutions is not greater than 1 or there was no couple of the refined solutions found during the previous step, we can conclude that additional investigations into the equation set are required.

Step 11. End of algorithm.

In order to check whether it's reasonable to use the proposed algorithm as a part of firmware for discarding inapplicable measurement results, the algorithm was implemented in Keil uVision for ARM-

based stm32f407vg microcontroller included into evaluation boards stm32f4discovery. The algorithm's implementation was verified using test cases comprised by 40 different sets of quadrics' equations with known solutions. Among the equation sets there were those containing the only intersection point, two intersection points, 4 intersection points and no intersection points at all. Besides, we've checked the situations where the existing intersection points resided out of the region where there have been sought for.

The algorithm detected all the solutions in just about 40% of test cases. There was no occasion when a region not containing solutions was claimed to contain them. It was found that the algorithm works about 6 times faster than the algorithm in [7].

Conclusions

The work introduced a simple algorithm for detecting whether a set of quadrics' equations has multiple solutions. The algorithm's implementation can be used in firmware of smart sensors of vector quantities. Such sensors can have output characteristics described by quadrics' equations and the coordinates of the vector being measured in this case should be found as a solution of the equation set (i.e. quadrics' intersection point). If there are several solutions, these measurement results are not applicable.

Verification of the algorithm implementation in stm32f407 microcontroller's firmware has shown low accuracy of finding the amount of solutions but quite acceptable time of execution. The very idea of the algorithm explains its fast work and risk to "miss" solutions. For this reason, if during execution of the algorithm less than two different intersections points of quadrics were found, it's not enough information to draw any conclusions about the actual amount of intersection points and, consequently, about applicability of the measurement results. In this case additional study of the equation set is needed. On the other hand, if two or more solutions were found, this fact proves inability to use the measurement results. Thus, the introduced algorithm can be used as a "smoke test" in order to perform a sketchy review of the equation set and is more reasonable for examining equation sets that actually have multiple solutions than the algorithm introduced in [7] despite the latter ensures that no region containing the solutions is missed.

1. Мікроелектронні сенсорні пристрої магнітного поля / І. А. Большакова, М. Р. Гладун, Р. Л. Голяка, З. Ю. Готра, І. Є. Лопатинський, Є. Потенцкі, Л. І. Сопільник; За ред. З. Ю. Готри. – Львів: Вид. Нац. ун-ту "Львівська політехніка", 2001. 2. Bolshakova, I. A. Methods of modeling and calibrating 3D magnetic sensors based on splitted Hall structures [Text] / I. A. Bolshakova, R. L. Holyaka, Z. Yu. Hotra, T. A. Marusenkova // Electronics and communications. Electronics and nanotechnologies. – 2011. – Vol. 2, Issue 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. Jouane // 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. Марусенкова Т. А. Метод оцінювання похибки вимірювання магнітного поля 3D-зондом з нелінійними вихідними характеристиками / Т. А. Марусенкова // Радіоелектроніка. Інформатика. Управління. – 2014. – № 2. – С. 38–43. 7. Марусенкова Т. А. Розробка алгоритму та мікропрограмного забезпечення для локалізації точок перетину квадрик / Т. А. Марусенкова, Д. О. Горман // Восточно-европейский журнал передовых технологий. – 2015. – № 75. – С. 16–20. 8. Кокс Д., Литтл Дж., О'Ши Д., Идеалы, многообразия и алгоритмы: Введение в вычислительные аспекты алгебраической геометрии и коммутативной алгебры. – М.: Мир, 2000. 9. Бухбергер Б. Алгоритмический метод в теории полиномиальных идеалов // Компьютерная алгебра. Символьные и алгебраические вычисления. – М.: Мир, 1986.