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

УДК 004.4, 519.688
DOI: 10.15587/1729-4061.2016.84889

РОЗРОБКА АЛГОРИТМУ З ПОКРАЩЕНОЮ РЕЛЕВАНТНІСТЮ ЛОКАЛІЗАЦІЇ КООРДИНАТ ВЕКТОРА ДЛЯ ІНТЕЛЕКТУАЛЬНИХ СЕНСОРІВ

Т. А. Марусенкова
Кандидат технічних наук, старший викладач
Кафедра програмного забезпечення
Національний університет
«Львівська політехніка»
вул. С. Бандери, 12, м. Львів, Україна, 79013
E-mail: tetyana.marus@gmail.com

1. Вступ

Сенсори векторних величин є важливою ланкою у зборі первинних даних для їхнього подальшого опрацювання. З метою підвищення чутливості сенсорів і, відтак, точності вимірювання, а також з метою зменшення їхніх габаритів і, отже, збільшення щільності картографування вимірюваної величини розробляються нові конструктивні рішення сенсорів векторних величин [1].

Однак покращення одних показників нерідко тягне за собою погіршання інших. В результаті існує ряд сенсорів векторних величин нового покоління, польові характеристики яких залежать від усіх трьох координат вимірюваної векторної величини, їхніх квадратів і попарних добутків [2]. Визначення координат шуканого вектора зводиться у цьому випадку до розв'язання системи рівнянь, що описують поверхні другого порядку (квадрики). Така ускладнена процедура опрацювання результатів вимірювання призводить до ідеї побудови інтелектуальних сенсорів векторних величин – тобто, для зручності використання сенсорів з нелінійними характеристиками до них додають «мозок», здатний виконувати необхідні обчислення [3–5]. В ролі «мозку» часто виступає мікроконтролер, що поступається комп'ютеру в тактовій частоті та обсягах постійної та оперативної пам'яті. Відповідно, до програмного забезпечення інтелектуальних сенсорів, як і будь-яких вбудованих систем, висуваються вимоги економного використання пам'яті програм і даних, а також зменшеної обчислювальної складності.

Оскільки аналітичні розв'язки можуть бути одержані лише для вузьких класів систем рівнянь, пошук розв'язків системи рівнянь передбачає локалізацію областей, де вони містяться, і їхнє подальше уточнення. Класичні алгоритми пошуку розв'язків систем рівнянь квадрик незастосовні для реалізації у програмному забезпеченні інтелектуальних сенсорів внаслідок їхньої чутливості до точності збереження результатів проміжних обчислень. Наявні алгоритми, орієнтовані на реалізацію в мікропрограмному забезпеченні, є недостатньо проробленими, і цей факт зумовлює необхідність у проведенні досліджень, спрямованих на покращення показників роботи цих алгоритмів, передусім, точністю, обчислювальною складністю та обсягами затраченої постійної і оперативної пам'яті.

2. Аналіз літературних даних і постановка проблеми

Одним з найпотужніших підходів до розв'язання поліноміальних рівнянь і їхніх систем є застосування базису Гребнера. У 1965 р. запропоновано алгоритм, який дозволяє за скінченну кількість кроків побудувати базис Гребнера ідеалу [6]. Перевагою алгоритму Бухбергера є універсальність – алгоритм застосовний для будь-якої системи поліноміальних рівнянь. Однак алгоритм Бухбергера орієнтований на символьні обчислення і є доволі повільним для великих систем, що знижує його застосовність на практиці. Рядом учених проводилися дослідження альтернативних, пришвидшених шляхів побудови базису Гребнера. У 1883 р.

запропоновано альтернативний алгоритм побудови базису Гребнера, швидший за алгоритм Бухбергера. Згодом, ідея цього алгоритму лягла в основу алгоритму F4 (1994 р.), що дав змогу уникнути надлишкових проміжних обчислень і був згодом реалізований у Maple. У 2002 р. запропоновано алгоритм F5, який вважається одним з найшвидших алгоритмів для побудови базису Гребнера [7]. Алгоритм F5 використовує розріджені матриці, які, як відомо, займають багато пам'яті. Алгоритм F5C, пришвидшена модифікація алгоритму F5, запропонована у [8]. Подальші вдосконалення алгоритму досягнуті у [9]. У [10] запропоновано G2V – наступну пришвидшену модифікацію F5. Паралельно з новими алгоритмами проводилися також паралельні обчислення класичного алгоритму Бухбергера. Аналіз останніх публікацій, присвячених розвитку застосування базису Гребнера для розв'язання поліноміальних рівнянь і їхніх систем свідчить про те, що ведуться дослідження у напрямку зменшення обчислювальної складності алгоритмів для його побудови, однак без втрати універсальності цих алгоритмів. Тим не менш, новітні модифікації алгоритмів побудови базису Гребнера усе ще не застосовні у мікропрограмному забезпеченні внаслідок їхньої складності, вимог до точності збереження проміжних результатів перетворень і затребуваної пам'яті. З іншого боку, застосування цих універсальних алгоритмів є надлишковим у випадках, коли потрібно лише розв'язати систему з трьох рівнянь поверхонь другого порядку. Таким чином, реалізація алгоритмів побудови базису Гребнера суперечить загальним принципам мінімального споживання ресурсів у програмному забезпеченні вбудованих систем.

Пошук та відображення перетину поверхонь другого порядку є задачею, що набула важливості з розвитком комп'ютерної графіки та систем автоматизованого проектування. В останніх здатність опрацьовувати криві та поверхні появилася з 60-их рр., однак, закладеним в них методикам бракувало або точності, або прийнятного часу виконання. Ряд публікацій присвячено проблематиці точного обчислення кривих перетину поверхонь і точок перетину кривих. Розроблено математичні засади для обчислення упорядкування довільних квадрик (зокрема, вироджених) у тривимірному просторі. Для цього задача зводиться до меншої розмірності, тобто до двовимірного випадку, що вирішується циліндричною алгебраїчною декомпозицією. Реалізація останньої сама по собі вимагає чимало обчислень і останнім часом проводилися роботи зі спрощення цього процесу, зокрема і за допомогою базису Гребнера [11]. У [12] вирішено задачу ефективного і точного параметричного представлення кривої перетину двох квадрик. Подальший розвиток ці результати дістали у [13], де запропоновано ефективний, точний і повний алгоритм побудови графа суміжності впорядкування квадрик.

Тим не менше, аналіз наявних публікацій в галузі обчислювальної геометрії показує, що увага вчених зосереджена на розвитку символьно-чисельних алгоритмів, які є недоцільними для застосування у мікропрограмному забезпеченні.

Існують бібліотеки, оптимізовані для різних архітектур мікроконтролерів, які реалізують чисельні методи для розв'язання довільних рівнянь, незалежно від їхньої природи. Як відомо, розв'язання нелінійних рівнянь чисельними методами включає відділення коренів (локалізацію розв'язків) і їхнє уточнення. У [14] досліджено вплив локалізації потенційних розв'язків системи нелінійних алгебраїчних рівнянь на успішність уточнення цих розв'язків застосовними для мікроконтролерів архітектури ARM бібліотечними функціями, які реалізують відомі чисельні методи (як правило, це метод дихотомії, метод Ньютона і метод хорд). Усі бібліотечні функції потребують наближеного розв'язку, який у вигляді чисел чи інтервалів передають їм у якості параметрів. У ході дослідження зазначених бібліотечних функцій виявлено, що їхня робота є нестабільною, якщо в якості параметра передано некоректно сформований інтервал (недостатньо вузький або такий, що містить декілька розв'язків рівняння) або недостатньо наближений розв'язок. Багатомірний метод Ньютона дозволяє уточнювати наближений розв'язок системи нелінійних рівнянь, однак, для будь-якого вектора початкових наближень метод зможе знайти лише один розв'язок, отже, для відшукання усіх розв'язків необхідно знати їхню кількість і області, в яких вони знаходяться. Усе це свідчить про важливість правильної локалізації розв'язків. В ідеальному випадку внаслідок етапу локалізації розв'язків системи рівнянь має бути знайдено стільки областей, скільки є розв'язків.

Тим не менш, на сьогодні немає готових бібліотек для локалізації розв'язків поліноміальних рівнянь і їхніх систем. Таким чином, актуальною є задача розроблення алгоритмів і їх реалізацій для локалізації розв'язків поліноміальних рівнянь і їхніх систем, які б відповідали вимогам до мінімального споживання ресурсів. Більше того, паралельно з дослідженнями, присвяченими розвитку універсальних алгоритмів, є потреба у дослідженнях, спрямованих на розроблення швидких і простих алгоритмів, які дозволяли б мінімальними затратами вирішувати певний вузький клас задач.

У [15] представлено алгоритм відділення коренів поліноміальних функцій однієї змінної і мікропрограмне забезпечення для контролерів сімейства ARM. У [16] описаний спосіб локалізації розв'язків системи рівнянь вигляду

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

за допомогою обчислення матриці Сильвестра. У [14] представлено алгоритм для локалізації поліноміальних рівнянь і їхніх систем, що застосовує інтервальну арифметику. Основна ідея алгоритму полягає в наступному. Усі рівняння в системі приводять до такого вигляду, щоб їхні праві частини були рівні 0. Вся область пошуку розв'язків системи рівнянь розбивається на суміжні підобласті у формі однакових за розмірами прямокутних паралелепіпедів

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

Замість точкових значень x, y і z у кожну з функцій, які представляють ліві частини рівнянь, підставляють інтервали $[x_i, x_{i+1}]$, $[y_j, y_{j+1}]$, $[z_k, z_{k+1}]$. В результаті одержуємо три інтервали. Якщо інтервал містить 0, значить, існує точка в розглядуваній області, в якій функція набуває значення 0, тобто, ця точка є розв'яз-

ком відповідного рівняння в системі. Інакше в даній області немає такої точки. Таким чином, якщо всі обчислені інтервали містять 0, область вважаємо такою, що потенційно містить розв'язки. Інакше область відкидаємо. Алгоритм є простим в реалізації і гарантовано знаходить усі області, які містять розв'язки, а його недоліком є те, що частина знайдених областей є «хибною тривогою», тобто, насправді не містить точок перетину квадрик. Тому є необхідність у пошуку альтернативних алгоритмів, які б забезпечили вищу релевантність результатів.

3. Мета та задачі дослідження

Метою роботи є удосконалення алгоритму локалізації розв'язків систем рівнянь квадрик для інтелектуальних сенсорів векторних величин на основі мікроконтролерів архітектури ARM Cortex-M за релевантністю знайдених областей з потенційними розв'язками. Якщо в удосконаленого за цим показником алгоритму погіршаться швидкість і споживання постійної і оперативної пам'яті, то таке погіршання повинно компенсуватися підвищенням релевантності знайдених областей.

Для досягнення поставленої мети були поставлені наступні завдання:

– проаналізувати та систематизувати математичний апарат, що може слугувати підґрунтям для удосконалення алгоритму локалізації точок перетину квадрик, розробити і реалізувати новий алгоритм для мікроконтролера на основі архітектури ARM Cortex-M4;

– провести порівняння реалізацій наявного та удосконаленого алгоритмів локалізації точок перетину квадрик за релевантністю знайдених за цими алгоритмами областей з потенційними точками перетину, а також часом виконання, обсягом коду і використанням оперативної пам'яті.

4. 1. Математичний апарат для локалізації точок перетину квадрик

Нехай інтелектуальний сенсор векторної величини містить три давачі, польові характеристики яких описуються рівняннями квадрик:

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

де x, y і z – проекції вимірюваної векторної величини на осі абсцис, ординат і аплікат деякої декартової системи координат, прив'язаної до корпусу сенсора, a_{ij}, b_{ij}, c_{ij} $(i=\overline{1,3},\ j=\overline{1,3})$ – відомі коефіцієнти польових характеристик давачів, S_i – виміряні сигнали кожного з трьох давачів.

Позначимо через f_1, f_2 і f_3, функції, що представляють ліві частини рівнянь (1):

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

Розглянемо детальніше поведінку функції f_1 у межах прямокутного паралелепіпеда

$$x_a\le x\le x_b,\ \ y_a\le y\le y_b,\ \ z_a\le z\le z_b.$$

Якщо у різних вершинах паралелепіпеда функція набуває різних знаків, то внаслідок її неперервності знайдеться така точка в межах вказаного прямокутного паралелепіпеда, в якій функція набуває значення 0. Якщо ж у всіх вершинах паралелепіпеда функція або додатна, або від'ємна, не можна зробити висновок, що в заданій області немає точки, де функція перетворюється на нуль, – потрібні додаткові перевірки.

Як відомо, максимального (і мінімального) значення в обмеженій області функція набуває або в точках екстремуму, або на границі. Таким чином, слід знайти максимальне значення на границі і порівняти його зі значеннями в точках екстремуму всередині області, якщо такі точки там є.

Границя паралелепіпеда – це шість його граней. На кожній з цих граней одна зі змінних функції f_1 фіксована, тобто, приходимо до 6 функцій від двох змінних, що у критичних точках набувають значення 0:

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

Кожна з шести функцій, що представляють ліву частину рівнянь (3), у свою чергу, набуває максимального значення або у критичних точках (якщо вони є в області, обмеженій відповідним прямокутником), або на границях.

Границями для функцій, що представляють ліву частину рівнянь (3), є ребра прямокутного паралелепіпеда $x_a\le x\le x_b,\ \ y_a\le y\le y_b,\ \ z_a\le z\le z_b$:

$$\begin{aligned}
&a_{11}x^2+a_{12}y_i^2+a_{13}z_j^2+b_{11}xy_i+b_{12}xz_j+\\
&+b_{13}y_iz_j+c_{11}x+c_{12}y_i+c_{13}z_j-S_1=0,
\end{aligned}$$

$$a_{11}x_k^2+a_{12}y^2+a_{13}z_j^2+b_{11}x_ky+b_{12}x_kz_j+$$
$$+b_{13}yz_j+c_{11}x_k+c_{12}y+c_{13}z_j-S_1=0,$$

$$a_{11}x_k^2+a_{12}y_i^2+a_{13}z^2+b_{11}x_ky_i+b_{12}x_kz+$$
$$+b_{13}y_iz+c_{11}x_k+c_{12}y_i+c_{13}z-S_1=0, \quad (4)$$

де $x_k\in\{x_a,x_b\}$, $y_i\in\{y_a,y_b\}$, $z_j\in\{z_a,z_b\}$ (тобто, для кожного рівняння є чотири комбінації пари фіксованих змінних).

Кожна з функцій, що представляють ліву частину рівнянь (4), набуває максимального (і мінімального) значення або у критичних точках (якщо вони є на відповідному ребрі розглядуваного паралелепіпеда), або на кінцях відрізку. Кінці відрізку – це вершини паралелепіпеда, представленого усіма точками

$$(x_k,y_i,z_j),\ x_k\in\{x_a,x_b\},\ y_i\in\{y_a,y_b\},\ z_j\in\{z_a,z_b\}.$$

Для пошуку критичних точок кожної з функцій, що представляють ліву частину рівнянь (3), знаходимо частинні похідні і прирівнюємо їх до нуля. Наприклад, для першої цих функцій:

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

критичні точки визначаються з системи лінійних алгебраїчних рівнянь:

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

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

Критичні точки решти функцій, що представляють ліву частину рівнянь (3), визначаються аналогічним способом.

Для знаходження точок екстремуму функцій, що представляють ліву частину рівнянь (4), візьмемо їхні похідні і прирівняємо їх до нуля:

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

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

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

Таким чином, кожна з функцій, що представляють ліву частину рівнянь (4), може мати не більше ніж одну критичну точку (якщо вона належить розглядуваному ребру), і критичні точки визначаються за формулами:

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

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

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

Отже, якщо потрібно перевірити існування точок перетину квадрик у деякій області, що має форму прямокутного паралелепіпеда $x\in[x_a,x_b]$, $y\in[y_a,y_b]$, $z\in[z_a,z_b]$, достатньо встановити знаки функцій (2) у всіх вершинах паралелепіпеда, функцій, що представляють ліву частину рівнянь (4), у точках (7), тобто, на ребрах паралелепіпеда, і функцій, що представляють ліву частину рівнянь (3), у точках, що є розв'язками систем лінійних алгебраїчних рівнянь вигляду (5). Якщо для якоїсь з функцій групи (2) виявлено зміну знаку (наприклад, функція від'ємна в усіх вершинах

$$(x_k,y_i,z_j),\ x_k\in\{x_a,x_b\},\ y_i\in\{y_a,y_b\},\ z_j\in\{z_a,z_b\},$$

але набуває невід'ємного значення в якійсь з точок на ребрі паралелепіпеда чи його грані), то в розглядуваній області існує точка, в якій ця функція набуває значення 0. Якщо зміну знаків виявлено для кожної з функцій (2), є ймовірність, що в розглядуваній області є точка перетину квадрик (1). Не виключено, що в межах цієї області усі квадрики перетинаються попарно, але не в одній точці, тому відповідь на питання про існування точки перетину може бути отримана лише після уточнення розв'язків за одним з відомих методів або ж у випадку, коли x_b-x_a, y_b-y_a та z_b-z_a є настільки малими числами, що середини діапазонів $[x_a,x_b]$, $[y_a,y_b]$, $[z_a,z_b]$ можуть з прийнятною похибкою вважатися розв'язком системи рівнянь (1). З іншого боку, відсутність зміни знаків хоча б для однієї з функцій (2) означає, що в розглядуваній області гарантовано немає точок перетину квадрик (1).

4. 2. Методика визначення областей точок перетину квадрик

Як і у [14], всю область D, в якій шукатимемо розв'язки системи рівнянь (2), розбиваємо на однакові за розмірами прямокутні паралелепіпеди: візьмемо L рівновіддалених точок на осі Ox, M – на осі Oy і N – на осі Oz.

На відміну від запропонованого у [14] підході до локалізації розв'язків системи трьох рівнянь поверхонь другого порядку, що базується на інтервальній арифметиці і передбачає виділення пам'яті для збереження обчислених границь інтервалів самих квадрик (1), діапазонів змінних $[x_k,x_{k+1}]$, $[y_i,y_{i+1}]$, $[z_j,z_{j+1}]$ та проміжних результатів обчислень, підхід, представлений у даній роботі, дозволяє зберігати лише дані про знаки функцій (2) у кожній з вузлових точок і функцій, які представляють ліві частини рівнянь (3) та (4), у тих їхніх критичних точках, визначених з (5) та (7), що належать ребрам і граням поточного розглядуваного прямокутного паралелепіпеда. Для збереження знаку достатньо одного біта (далі за текстом вважатимемо, що 1 кодує від'ємне значення, а 0 – невід'ємне), що дозволяє розробити і реалізувати алгоритм локалізації точок перетину квадрик із застосуванням змінних беззнакового цілого типу замість змінних дійсного типу. При цьому у відведених змінних використовуватиметься кожен біт на противагу до ситуації, коли у змінній частина біт зайнята даними, а решта біт – доповнюючи нулі.

Для зберігання знаків функції f_1 у вузлових точках створимо масив F1_Nodes розмірів $M\times N$ беззнакового цілого типу T, що вміщає L біт. Кожен елемент F1_Nodes[i][j] цього масиву представлятиме значення функції f_1 у точках, в яких $y=y_i$, $z=z_j$, а x

набуває значення x_k $(k=\overline{1,L})$, причому біт з номером k (у порядку LSB) відповідатиме знаку функції f_1 у точці (x_k,y_i,z_j). Наприклад, якщо M=N=8 і T – тип, що представляється 8 бітами, то F1_Nodes[i][j]=10010011 означатиме, що функція f_1 у точках (x_1,y_i,z_j), (x_4,y_i,z_j), (x_7,y_i,z_j), (x_8,y_i,z_j) набуває від'ємних значень, а в точках (x_2,y_i,z_j), (x_3,y_i,z_j), (x_5,y_i,z_j), (x_6,y_i,z_j) – невід'ємних.

Для збереження даних про знаки функцій f_2 і f_3 у тих же вузлових точках потрібні два інші масиви (F2_Nodes[i][j] і F3_Nodes[i][j]) тих же розмірів і того ж типу. Усього ці три масиви займатимуть $3L\times M\times N/8$ байт.

Для відстеження наявності критичних точок і знаків похідної кожної з трьох функцій (2) створюємо 6 масивів: F 1_PosExtremums, F1_NegExtremums, F2_PosExtremums, F2_NegExtremums, F3_PosExtremums і F3_NegExtremums. Для кожної комбінації x та y, x та z і y та z, знаходимо ті єдині значення z, y і x відповідно, що для них рівняння (4) перетворюються на рівності. Далі, визначаємо, яким діапазонам $[z_j,z_{j+1}]$ $(j=\overline{1,N-1})$ $(i=\overline{1,M-1})$ $[x_k,x_{k+1}]$ $(k=\overline{1,L-1})$ належать знайдені значення z, y і x відповідно. Достатньо запам'ятовувати лише початок кожного з визначених діапазонів (оскільки довжина кожного діапазону є відомою). Кожне ребро (і критична точка на ньому) може належати чотирьом або двом суміжним паралелепіпедам або ж лише одному паралелепіпеду. У біті з номером k елемента масиву F1_PosExtremums[i][j] зберігатимемо 1, якщо хоча б на одному з 12 ребер паралелепіпеда, одна з вершин якого є точкою (x_k,y_i,z_j) і саме ця вершина розташована найближче до початку координат, є критична точка, де функція, що представляє ліву частину першого рівняння з системи рівнянь (4), набуває невід'ємного значення. Призначення масиву F1_NegExtremums є аналогічним з тією різницею, що в ньому зберігаємо 1 у випадку, коли функція набуває від'ємного значення принаймні в одній критичній точці на ребрах паралелепіпеда з вершиною (x_k,y_i,z_j). Нулі у бітах елементів цих масивів кодуватимуть відсутність критичних точок у відповідних паралелепіпедах. Чотири інші масиви потрібні для зберігання аналогічних даних про критичні точки функцій f_2 і f_3 на ребрах паралелепіпедів.

Отже, коли знаходимо критичну точку (x,y_i,z_j), $x\in[x_k,x_{k+1}]$ функції, що представляє ліву частину першого рівняння з системи рівнянь (4), слід виконати наступні дії:

– встановити знак функції у ній;

– в масиві F1_NegExtremums або F1_PosExtremums (в залежності від знаку функції) встановити у 1 наступні біти:

– біт з номером k елемента з індексами i та j;

– біт з номером k елемента з індексами (i–1) та j, якщо виконується умова i>0 та i<M–1;

– біт з номером k елемента з індексами i та (j–1), якщо виконується умова j>0 та j<N–1;

– біт з номером k елемента з індексами (i–1) та (j–1), якщо виконуються умови (i>0 та i<M–1) і (j>0 та j<N–1).

Для критичних точок, що знаходяться на ребрах, паралельних осі Oy і осі Oz, підхід залишається той самий, змінюються лише аналізовані індекси.

Далі, слід визначити знак максимального/мінімального значення кожної з функцій (2) на гранях паралелепіпедів. Для цього знаходимо розв'язок системи лінійних алгебраїчних рівнянь вигляду (5) з двома невідомими для кожної з функцій (2) і кожного значення значення x_k, y_i і z_j $(k=\overline{1,L},\ i=\overline{1,M},\ j=\overline{1,N})$. На відміну від критичних точок на ребрах, для критичних точок на гранях потрібно визначити діапазони, в які потрапляє пара знайдених значень – y та z для площин $x=x_k$, x та z для площин $y=y_i$ і x та y для площин $z=z_j$.

Кожна грань може належати тільки одному або двом суміжним паралелепіпедам. Тому за аналогією з критичними точками (7), відносимо знайдені критичні точки до паралелепіпедів, у кожного з яких вершина з найменшими координатами є однією з вузлових точок. Для зберігання знаків функцій на гранях паралелепіпедів можна використати вже наявні масиви, визначені раніше для знаків функцій у критичних точках (7). Якщо критична точка належить паралелепіпеду, для якого точка x_k, y_i і z_j $(k=\overline{1,L-1},\ i=\overline{1,M-1},\ j=\overline{1,N-1})$ є вершиною з найменшими координатами, а саме, його грані, перпендикулярній осі Ox, то слід виконати дії:

– визначити знак функції у критичній точці;

– в один з масивів (XX_PosExtremums або XX_NegExtremums) в залежності від знаку функції, встановити в 1 біти у:

– біт з номером k елемента з індексами i та j;

– біт з номером k елемента з індексами (i–1) та j, якщо виконується умова i>0 та i<M–1.

Для площин, перпендикулярних осям Oy і Oz, підхід залишається тим самим (міняються лише індекси).

Загальний обсяг пам'яті для масивів F1_PosExtremums, F1_NegExtremums, F2_PosExtremums, F2_NegExtremums, F3_PosExtremums і F3_NegExtremums становить $6L\times M\times N/8$ байт.

Коли сформовані всі масиви, можна починати аналізувати знаки функцій у всіх вузлових і критичних точках. Для збереження результатів доцільним є створення масиву RootsLocalized тих же розмірів, що й кожен з масивів, визначених раніше. Кожен елемент цього масиву визначаємо наступним чином: порівнюємо знаки функції f_1 у кожній вершині паралелепіпеда (x_k,y_i,z_j) зі знаками цієї ж функції у сусідніх вершинах

(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}), $(x_{k+1},y_{i+1},z_{j+1})$.

Ця умова виглядатиме так:

```
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)
```

Якщо з-поміж восьми вершин знайдуться такі, в яких функція набуває різні знаки, то результатом цього виразу буде 1, інакше – 0. Результатом порівняння (NOT(F1_Nodes[i][j])) AND F1_NegExtremums[i][j] буде 1 лише нульових бітів у елементі масиву F1_Nodes[i][j], тобто, для тих з L описаних цим елементом вершин, в яких функція має невід'ємний знак

(і відповідно, знайдено зміну знаків). Результатом порівняння (F1_Nodes[i][j] AND F1_PosExtremums[i][j]) буде 1 лише для тих бітів F1_Nodes[i][j], що представляють вершини, де функція набуває від'ємних значень, тоді як одиниця у відповідних бітах масиву F1_PosExtremums[i][j] означає невід'ємний знак функції. Загальна умова зміни знаку функції у вершинах або критичних точках виглядатиме як:

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

Неважливо, що кожен біт у масиві F1_Nodes задає лише одну вершину паралелепіпеда, а не всі вісім, оскільки якщо серед восьми вершин є вершини з різним знаком, значення виразу буде 1, а інакше – всі вершини мають однаковий знак і неважливо, знак якої з них порівнюємо.

Аналогічним чином формуємо значення V2 і V3. Паралелепіпед може містити точки перетину квадрик (1), якщо

V1 AND V2 AND V3 (10)

є істиною.

4. 3. Алгоритм локалізації точок перетину трьох квадрик

1. Визначити замкнуту область D, в якій шукатимемо розв'язки. У загальному випадку, підставою для вибору можуть бути знання про діапазон вимірювання. У цьому випадку D матиме форму куба, оскільки кожна координата векторної величини може перебігати будь-які значення від нуля до модуля шуканої векторної величини (але без втрати загальності казатимемо про прямокутний паралелепіпед). Наприклад, якщо модуль вимірюваної величини може перебувати в діапазоні від V_{min} до V_{max}, то кожну з координат шукатимемо в області від $-V_{max}$ до V_{max}. Якщо з-поміж заданих квадрик є обмежені поверхні (наприклад, еліпсоїди), область D може бути визначена за рахунок визначення, до якого з 17 типів квадрик належить кожна з квадрик (1), і приведення рівняння квадрики до канонічного вигляду.

2. Задати L, M і N точок, на які розбиті координатні осі Ox, Oy і Oz, відповідно. Для зручності роботи з бітами і масивами елементів скалярних типів кожне з цих чисел не повинно перевищувати кількість біт у беззнаковому цілому типі даних (з міркувань, викладених вище).

3. Створити масиви F1_Nodes, F2_Nodes, F3_Nodes, F1_PosExtremums, F1_NegExtremums, F2_PosExtremums, F2_NegExtremums, F3_PosExtremums, F3_NegExtremums і RootsLocalized та ініціалізувати їх нулями.

4. Для кожної комбінації значень x_k, y_i і z_j ($k=\overline{1,L}$, $i=\overline{1,M}$, $j=\overline{1,N}$) за допомогою трьох циклів слід:

4. 1. знайти значення кожної з функцій (2) і встановити біт з номером k з індексами i та j у відповідному масиві (F1_Nodes, F2_Nodes або F3_Nodes), якщо функція від'ємна у точці , інакше – 0;

4. 2. обчислити за формулами (7) значення x, y і z для кожної з трьох функцій такі, що в точках (x,y_i,z_j), (x_k,y,z_j) і (x_k,y_i,z) рівняння (4) перетворюються на рівності, а також обчислити знаки функцій (2) у цих точках;

4. 3. знайти значення k_1, i_1, j_1 такі, що

$$x\in\left[x_{k1},x_{k1+1}\right],\ \ y\in\left[y_{i1},y_{i1+1}\right]\ \text{і}\ z\in\left[z_{j1},z_{j1+1}\right],$$

тобто, визначити, яким ребрам паралелепіпедів належать точки екстремуму, обчислені на кроці 4. 2;

4. 4. встановити в 1 біт з номером k_1 елемента з індексами i та j, біт з номером k елемента з індексами i_1 та j і біт з номером k елемента з індексами i та j_1 у відповідному масиві (одному з F1_PosExtremums, F1_NegExtremums, F2_PosExtremums, F2_NegExtremums, F3_PosExtremums і F3_NegExtremums, в залежності від аналізованої функції і її знаку);

4. 5. встановити біти для всіх «сусідів» (паралелепіпедів, що теж містять знайдені ребра); їх усього може бути 9, 3 або 0 у відповідності до наведених вище міркувань;

4. 6. знайти такі комбінації значень y_1 і z_1, x_1 і z_2, x_2 і y_2 для кожної функції, що у точках

$$(x_k,y_1,z_1),\ \ (x_1,y_i,z_2)\ \text{і}\ (x_2,y_2,z_j)$$

рівняння (5) перетворюються на рівності;

4. 7. знайти значення i_1 та j_1, k_1 та j_2 і k_2 та i_2 такі, що

$$y_1\in\left[y_{i1},y_{i1+1}\right]\ \text{і}\ z_1\in\left[z_{j1},z_{j1+1}\right],$$

$$x_1\in\left[x_{k1},x_{k1+1}\right]\ \text{і}\ z_2\in\left[z_{j2},z_{j2+1}\right],$$

$$x_2\in\left[x_{k2},x_{k2+1}\right]\ \text{і}\ y_2\in\left[y_{i2},y_{i2+1}\right].$$

Дізнатися знаки функцій (2) у точках, знайдених на кроці 4. 6, і встановити одиниці у відповідних бітах масивів для відстеження знаків функцій у точках екстремуму за схемою, аналогічною описаній у кроці 4. 4;

4. 8. встановити біти для паралелепіпедів з гранями, визначеними на кроці 4. 7 (таких паралелепіпедів може бути 3 або 0).

5. Сформувати елемент RootsLocalized[i][j] з використанням (8)–(10) для кожної комбінації індексів i та j.

6. Обчислити критичні точки функцій (2), визначити проміжки, яким вони належать, і доповнити сформований масив RootsLocalized даними про знаки функцій (2) у критичних точках.

7. Кінець алгоритму.

5. Результати порівняльного аналізу запропонованого алгоритму з алгоритмом на основі інтервальної арифметики

Для з'ясування доцільності запропонованого алгоритму виконано ряд чисельних експериментів реалізацій двох алгоритмів – запропонованого у даній роботі та алгоритму на основі інтервальної арифметики [14].

Алгоритм був реалізований мовою C у середовищі Keil uVision 5, а тестування його виконувалося на мікроконтролері STM32F407VG (у складі оціночної плати STM32F4Discovery), побудованого на основі архітектури ARM Cortext-M4. Алгоритм на основі інтервальної арифметики [14] реалізований тією ж

мовою з використанням того ж компілятора і того ж мікроконтролера, що означає можливість коректного порівняння цих двох алгоритмів.

Порівняння реалізацій алгоритмів проводилося за наступними ознаками: частка областей, що були знайдені в результаті роботи алгоритму, але насправді не містили точок перетину квадрик, час виконання, обсяг машинного коду та використання оперативної пам'яті.

В якості тестових наборів було взято 48 систем з трьох рівнянь квадрик, точки перетину яких є відомими. Ручним тестуванням виявлено, що жоден з розв'язків систем рівнянь не був «пропущений» – для кожної з тестових систем рівнянь квадрик з-поміж результатів роботи алгоритму були області (прямокутні паралелепіпеди), що містили точки перетину квадрик. Однак, як і у випадку з алгоритмом, наведеним у [14], деякі зі знайдених областей виявилися «хибною тривогою» – насправді вони не містили розв'язків тестових систем рівнянь.

Спочатку перевірялися найпростіші варіанти систем рівнянь, для яких нескладно знайти аналітичні розв'язки – системи канонічних рівнянь квадрик (без лінійного переносу та повороту осей). У таких рівняннях частина коефіцієнтів при змінних, їхніх квадратах і попарних добутках є нульовими, що дозволило перевірити гілки коду, пов'язані з діленням на нуль. Порівняння областей, знайдених в ході роботи двох алгоритмів, виконувалося на однакових тестових наборах при тих самих вузлових точках (інакше результати порівняння не можна було б вважати коректними). Результати згруповані у табл. 1.

Таблиця 1

Кількості областей точок перетину квадрик, локалізованих реалізаціями двох алгоритмів

№	Система рівнянь квадрик	Розв'язки	Кількість областей, знайдена за алгоритмом на основі інтервальної арифметики		Кількість областей, знайдена за запропонованим алгоритмом	
			L = 8 M = 8 N = 8 (343 обл.)	L = 16 M = 16 N = 16 (3375 обл.)	L = 8 M = 8 N = 8 (343 обл.)	L = 16 M = 16 N = 16 (3375 обл.)
1	2	3	4	5	6	7
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	$\begin{cases} x^2+y^2+z^2-2=0 \\ x^2+0.05y^2+z^2-2=0 \\ x^2-2y^2+z^2-2=0 \end{cases}$	множина всіх точок $(x, 0, z)$, таких що $x^2+y^2=2$	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	$\begin{cases} x^2+0.05y^2+z^2-2=0 \\ x^2+y^2+z^2-2=0 \\ x^2+z^2+y^2-1=0 \end{cases}$	немає точок перетину	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

Продовження таблиці 1

1	2	3	4	5	6	7
9	$\begin{cases} x^2+y^2-2z=0 \\ x^2+y^2+z^2-2z-4=0 \\ x^2+z^2-4z+3=0 \end{cases}$	$\left(-1,-\sqrt{3},2\right),\ \left(-1,\sqrt{3},2\right),$ $\left(1,-\sqrt{3},2\right),\ \left(1,\sqrt{3},2\right)$	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	$\begin{cases} x^2+y^2-2z=0 \\ x^2+y^2+z^2-2z-4=0 \\ y^2+z^2-4z+2=0 \end{cases}$	$\left(-\sqrt{2},-\sqrt{2},2\right),\ \left(\sqrt{2},-\sqrt{2},2\right),$ $\left(-\sqrt{2},\sqrt{2},2\right),\ \left(\sqrt{2},\sqrt{2},2\right)$	46	58	24	24
12	$\begin{cases} x^2+y^2+z^2-4=0 \\ x^2-y^2+z^2=0 \\ x^2+y^2-z^2=0 \end{cases}$	$\left(0,-\sqrt{2},-\sqrt{2}\right),\ \left(0,-\sqrt{2},\sqrt{2}\right),$ $\left(0,\sqrt{2},-\sqrt{2}\right),\ \left(0,\sqrt{2},\sqrt{2}\right)$	46	58	16	16

Далі, до цих же систем рівнянь застосовувалися поворот осей і лінійний перенос.

Оскільки до кожного рівняння квадрики у системі застосовувалося однакове перетворення, кількість точок перетину зберігалася – змінювалися лише їхні координати у «старій» системі координат. До кожної з 12 тестових систем рівнянь, представлених у табл. 1, поворот виконувався на три різні комбінації кутів. Результати тестування виявилися приблизно такими ж, що й у табл. 1, з можливими відмінностями не більше, ніж на декілька одиниць.

Асимптотична оцінка складності обох алгоритмів у нотації «О велике» є однаковою, проте характер інструкцій різний, тобто, набір машинних інструкцій істотно відрізнятиметься, отже, може істотно відрізнятися і фактичний час виконання.

Оцінювання часу виконання реалізацій двох алгоритмів при однаковій кількості вузлових точок (тобто, при однакових наборах чисел L, M і N) здійснювалося лише практичним способом – за допомогою замірів часу роботи реалізації кожного з алгоритмів (при тій ж тактовій частоті) на тих же тестових наборах. Запропонований у даній роботі алгоритм виявився повільнішим у середньому на 12 %.

Обсяг машинного коду і задіяної оперативної пам'яті оцінювався за даними, взятими з map-файлу, згенерованого середовищем Keil uVision після компіляції та лінкування. Обсяг постійної пам'яті, затраченої алгоритмом, запропонованим у даній роботі, виявився на 20 % більшим, ніж аналогічний показник для алгоритму на основі інтервальної арифметики. Натомість обсяг використаної оперативної пам'яті виявився на 15 % меншим.

Доцільність використання алгоритму на основі інтервальної арифметики як проміжної ланки у процесі розв'язування систем поліноміальних рівнянь вже було доведено у [14]. Тестові набори для верифікації запропонованого в даній роботі алгоритму включали більшість систем рівнянь квадрик, на яких тестувався алгоритм у [14], і перевірка проводилася зокрема і при тій же області пошуку розв'язків і тих же вузлових точках. З цієї причини роль запропонованого алгоритму в уточненні розв'язків у даній роботі не досліджувалася.

6. Обговорення результатів порівняльного аналізу запропонованого алгоритму з алгоритмом на основі інтервальної арифметики

Результати тестування реалізацій двох порівнюваних алгоритмів дозволяють стверджувати, що новим алгоритмом досягнуто зменшення кількості нерелевантних областей з потенційними точками перетину трьох квадрик у приблизно 2,2 разів. Варто відзначити, щоправда, що кількість областей залежить від діапазонів $\left[x_a,x_b\right]$, $\left[y_a,y_b\right]$, $\left[z_a,z_b\right]$, при цьому звуження діапазонів не завжди призводить до збільшення (чи, навпаки, зменшення) кількості областей. Крім того, кількість «паразитних» областей виявилася різною для різних тестових наборів. Залежність «паразитних» областей від системи рівнянь квадрик пояснюється самим принципом роботи алгоритму – гарантовано «відсіюються» лише ті області, в яких хоча б одна пара квадрик не має точок перетину. Тобто, усі області, які містять точки перетину кожної пари квадрик, неодмінно вносяться у список областей, що потенційно можуть містити шукані розв'язки системи рівнянь квадрик. Однак, цілком можливо, що в деякій області всі квадрики перетинаються попарно, але не в одній спільній точці. Той факт, що алгоритм на основі інтервальної арифметики дає більший процент «паразитних» областей, узгоджується з результатами застосування інтервальної арифметики до оцінки похибок вимірювання – вони завжди є завищеними і вказують на «найгірший випадок». Поведінка алгоритмів досліджувалася для тестових випадків з однією, двома, чотирма та безліччю точок перетину трьох квадрик, а також за відсутності точок перетину. Обмеженість тестових наборів дає змогу говорити лише про приблизні відсотки виграшу в релевантності результатів запропонованого алгоритму по відношенню до алгоритму на основі інтервальної арифметики.

Тим не менш, проведені чисельні експерименти дозволяють стверджувати, що у випадку необхідності розв'язати систему рівнянь квадрик запропонований алгоритм є доцільнішим за його аналоги для реалізації у мікропрограмному забезпеченні, враховуючи обмеженість їхніх ресурсів. Одержані результати орієнто-

вані передусім на застосування в інтелектуальних сенсорах векторних величин, однак їх можна поширити на інші задачі, де фігурують рівняння поверхонь другого порядку та їхні системи. Більше того, та ж методика може бути застосована для будь-яких функцій трьох змінних, знаходження частинних похідних яких має таку ж складність, що й для функцій, які представляють ліві частини рівнянь (1). Якщо ж обчислення частинних похідних вимагає більше ресурсів (наприклад, у випадку рівнянь вищого степеню) або ж не існує аналітичного способу знаходження критичних точок функцій, що представляють ліві частини розв'язуваної системи рівнянь, потрібні додаткові дослідження.

Як відзначалося вище, новий алгоритм вимагає більше постійної пам'яті та виконується довше. Наведені числові показники не є абсолютними, оскільки стосуються лише однієї моделі мікроконтролера. Узагальнені результати можна одержати теоретично, якщо врахувати кількість інструкцій в алгоритмі і час виконання кожної з інструкцій для певної архітектури мікроконтролера.

Подальшим напрямом досліджень є зменшення обчислювальної складності запропонованого алгоритму. Потенційним шляхом досягнення такого результату є розбиття області на проміжки з різним кроком, на противагу поділу на проміжки однакової довжини.

7. Висновки

1. Здійснені аналіз і систематизація математичного апарату для локалізації точок перетину квадрик дозволили розробити новий алгоритм локалізації точок перетину квадрик, що базується на властивостях неперервних диференційовних функцій у замкнутих областях і перевірці знаків функцій у вузлових і критичних точках. Розроблений алгоритм реалізовано для мікроконтролера архітектури ARM Cortex-M4.

2. Проведений порівняльний аналіз нового алгоритму з алгоритмом-аналогом дозволяє стверджувати, що новий алгоритм забезпечує приблизно у 2.2 рази вищу релевантність результатів знайдених областей з потенційними точками перетину при на 12 % тривалішому виконанні. Новий алгоритм споживає на 20 % більше постійної пам'яті, зате на 15 % менше оперативної пам'яті. Таким чином, проведені дослідження довели доцільність застосування запропонованого алгоритму.

Література

1. Большакова, І. А. Мікроелектронні сенсорні пристрої магнітного поля [Текст] / І. А. Большакова, М. Р. Гладун, Р. Л. Голяка, З. Ю. Готра, І. Є. Лопатинський, Є. Потенцкі, Л. І. Сопільник; ред. З. Ю. Готра. – Львів: Вид. Національного університету «Львівська політехніка», 2001. – 412 с.
2. Большакова, І. А. Методи моделювання та калібрування 3D-зондів магнітного поля на розщеплених холлівських структурах [Текст] / І. А. Большакова, Р. Л. Голяка, З. Ю. Готра, Т. А. Марусенкова // Електроніка та зв'язок. Тематичний випуск «Електроніка та нанотехнології». – 2011. – № 2 (61). – С. 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 Gr bner 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. Марусенкова, Т. А. Розробка алгоритму та мікропрограмного забезпечення для локалізації точок перетину квадрик [Текст] / Т. А. Марусенкова, Д. О. Горман // Восточно-Европейский журнал передовых технологий. – 2015. – Т. 3, № 4 (75). – С. 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. Горман, Д. О. Алгоритм відділення коренів поліноміальних функцій для мікроконтролерів на основі ARM [Текст] / Д. О. Горман, Т. А. Марусенкова, І. Ю. Юрчак // XIII Міжнародна науково-технічна конференція CADSM "Досвід розробки та застосування приладо-технологічних САПР в мікроелектроніці". – Львів-Поляна, 2015. – С. 50–52.