DOI: 10.36558/rsc.v11i3.7391

Polynomial Algorithm for Quadratic Forms Classification Using Gaussian Elimination

Jesmmer da S. Alves
Instituto Federal Goiano - Campus Morrinhos
Núcleo de Computação
Morrinhos - Goiás, Brazil
jesmmer.alves@ifgoiano.edu.br

Diane Castonguay
Universidade Federal de Goiás
Instituto de Informática
Goiânia - Goiás, Brazil
diane@inf.ufg.br

ABSTRACT

Existing methods for quadratic forms classification have exponential time complexity or use approximation that weaken the result reliability. We developed an algorithm that improves the best case of quadratic form classification in constant time and is polynomial in the worst case. In addition, new strategies were used to guarantee the results reliability, by representing rational numbers as a fraction of integers and to identify linear combinations that are linearly independent using Gaussian Elimination.

CCS Concepts

•**Mathematics of computing** → **Nonlinear equations;**
•**Theory of computation** → **Quadratic programming;**

Keywords

quadratic forms; eigen problems; Gaussian Elimination; algorithm complexity

1. INTRODUCTION

The quadratic forms have applications in many different areas in computer science, mathematics, quantum physics, statistics and others (see some applications in [2, 3, 8, 1, 4, 16, 17, 11]). Here we deal with rational quadratic forms. For such quadratic forms, there exist a variety of methods and algorithms that allow, in some cases in polynomial time, to determine the type of the quadratic form. These are generally divided into direct methods, expansive methods and iterative methods.

The quadratic form may be classified as positive definite, negative definite, positive semidefinite, negative semidefinite and indefinite. The time complexity of algorithms to determine the classification of a quadratic form depends on the method and can be influenced by specific cases. Direct methods in general are exponential for any quadratic form, however, are polynomial for positive definite or negative definite quadratic forms. Expansive methods are not accurate because of lower/upper bound approximation. The time complexity of iterative methods is related to the desired approximation.

A problem comes from the fact that iterative methods use mathematical operations that break down the entries, showing a result with a large number of decimal places (see some resulting problems in [12, 13]). However, new strategies can be used to avoid this fractionation and provides a reliable result, such as: to store every matrix entry as the quotient of two integer; and use Gaussian Elimination to easy identify linear combinations of variables that are linearly independent.

The paper is organized as follows: in Section 2, we describe the quadratic forms basics concepts and the time complexity of known algorithms; in Section 3, we present some strategies that can be used in the quadratic form classification and make the result more reliable; in Section 4, we introduce a solution that makes the quadratic forms classification in polynomial time, it was divided in two algorithms: one that improves the best case of quadratic form classification in time $O(1)$; and one that evaluate all other cases in polynomial time. In Section 5, the final considerations are presented focusing on the analysis of the results achieved.

2. QUADRATIC FORMS

2.1 About quadratic forms

A quadratic form related to a symmetric matrix[1] $A_{n\times n}$ and evaluated in a vector $x_{n\times 1}$ is defined as [18]:

$$Q(x) = (x_1, \cdots, x_n) \cdot \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \vdots & \vdots \\ a_{n1} & \dots & a_{nn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \sum_{1\le i\le j\le n} a_{ij} x_i x_j \quad (1)$$

Every quadratic form can be represented by a symmetric matrix. Given a non-symmetric matrix B, the symmetrization of B is given by $A = \frac{1}{2}(B + B^t)$. Therefore, A is symmetric and $Q(x) = x^t Ax = x^t Bx$. The quadratic forms have a critical point, when $x = 0$, which is $Q(x) = x^t Ax = 0$. Therefore, all quadratic forms may be classified as the resulting polynomial of the expression $Q(x) = x^t Ax$, such as positive definite, negative definite, positive semidefinite, negative semidefinite and indefinite.

[1]Here, we will focus on symmetric matrix of rational values.

A quadratic form is **positive definite** (respectively **negative definite**) when $Q(x) = x^tAx > 0$ (respectively $Q(x) = x^tAx < 0$) for all $x \neq 0$. When $Q(x) = x^tAx \geq 0$, the quadratic form is **positive semidefinite**. Likewise, when $Q(x) = x^tAx \leq 0$, the quadratic form is **negative semidefinite**. Finally, the quadratic form is **indefinite** if $Q(x) = x^tAx > 0$ for some values of x and $Q(x) = x^tAx < 0$ for other values of x. The reader can see more about quadratic forms in [1, 18].

2.2 Quadratic forms classification

One way to verify whether a quadratic form is positive definite is through its naturally ordered principal minors (see more details in [6]). In general, this technique have cost $O(n^4)$ mainly influenced by operations like matrix multiplications.

Direct expansion is a more complete method for identifying all quadratic forms types [10]. It can be done through the principal minors of the matrix. Let A be a matrix of order n. A **principal submatrix** of order k of A is obtained by deleting $n-k$ columns and the corresponding $n-k$ lines of A. The determinant of a principal submatrix of order k is called **principal minor** of order k of A. In this case, the number of determinants of various orders is exponential.

The classification of quadratic forms also can be done by identification of the matrix characteristic polynomial, and then calculating limits to the roots of this polynomial (for more details see [14, 20]). The characteristic polynomial of a symmetric matrix $A_{n\times n} = (a_{ij})$ is the determinant $det(A - \lambda I)$ in λ of degree n.

Find the roots of a polynomial can be described as abstract math problem to find all the eigenvalues λ that satisfy these equation. After finding the characteristic polynomial, lower and upper bounds can be used to determine the range that contains all the roots of this polynomial. Despite being quite attractive, it makes the classification of quadratic forms not accurate, just because the existing methods return a lower/upper bound approximation and not a tight lower/upper bound.

Another option is to make use of iterative methods, such as algorithm *QR*, *Jacobi* and *Cholesky* decomposition [7, 15, 22]. Most of iterative methods have polynomial time complexity. However, due to the difficulty in finding exact eigenvalues, iterative methods are not stable and usually are implemented with an approximation to the number of eigenvalues decimal places (rounding and cancellations errors). Although some new strategies are more stable (see [19, 9]), in situations where cancellation errors can be very serious in orthogonalization steps, it is necessary to consider even more stable methods.

3. NEW STRATEGIES

3.1 Rational numbers as a fraction of Integers

Some operations, like division and radical, can increase the number of decimal places and make the result not reliable, or difficult to be analyzed. Fortunately, rational numbers can be represented like a fraction of integers. For this purpose, each entry of the matrix[2] representing the quadratic form can be expressed as the quotient of two integers.

Next are described Algorithms 1-4 to perform addition, subtraction, division and multiplication using fraction of integers. In this case, a, b and r are arrays with two positions (numerator and denominator). The function GCD returns the greatest common divisor between two integers and is used to reduce the numbers to the lowest value to be stored in the matrix [21]. A zero should be represented by $[0, 1]$ and the minus sign always accompanies the numerator (e.g.: $[-1, 2]$). It is assumed that $a[1]$ and $b[1]$ are nonzero.

Algorithm 3 uses Algorithm 4 to divide a by b and get as a result a rational number represented like a fraction of integers.

Algorithm 1: SUM(a, b)

Input: Arrays a and b.
Output: $r = a + b$.

```
1 r[1] ← a[1] * b[1]
2 r[0] ← a[0] * b[1] + b[0] * a[1]
3 gcd ← GCD(r[0], r[1])
4 r[0] ← r[0]/gcd
5 r[1] ← r[1]/gcd
6 return r
```

Algorithm 2: SUB(a, b)

Input: Arrays a and b.
Output: $r = a - b$.

```
1 r[1] ← a[1] * b[1]
2 r[0] ← a[0] * b[1] − b[0] * a[1]
3 gcd ← GCD(r[0], r[1])
4 r[0] ← r[0]/gcd
5 r[1] ← r[1]/gcd
6 return r
```

Algorithm 3: DIV(a, b)

Input: Arrays a and b.
Output: $r = \frac{a}{b}$.

```
1 b2[0] ← b[1]
2 b2[1] ← b[0]
3 r ← MULT(a, b2)
4 return r
```

Algorithm 4: MULT(a, b)

Input: Arrays a and b.
Output: $r = a * b$.

```
1 r[0] ← a[0] * b[0]
2 r[1] ← a[1] * b[1]
3 gcd ← GCD(r[0], r[1])
4 r[0] ← r[0]/gcd
5 r[1] ← r[1]/gcd
6 return r
```

[2]Most methods to make the classification of quadratic forms uses matrices.

3.2 Classification using Gaussian Elimination

Berger M. [5] has shown that for a quadratic form Q with n variables, there are linear combinations of variables l_i linearly independent and numbers c_i such that $Q = \sum_{i=1}^{n} c_i l_i^2$, for $i \leq n$.

Gaussian Elimination is a method to write a polynomial of degree 2 in n variables as perfect squares [16]. Thus, the type of a given quadratic form can be identified by Gaussian Elimination according to the number of perfect squares affected by sign "+" (positive) and sign "-" (negative), as follows: positive definite, if all squares are positive; negative definite, if all squares are negative; positive semidefinite (negative semidefinite), if some squares are positive (negative) and others null; and indefinite, if some squares are positive and others are negative.

Let Q be a nonzero quadratic form. The application of Gaussian Elimination may be carried out as follows:

- If $a_{ii} \neq 0$ (for some $i \in \{1, \ldots, n\}$), eliminate one variable. For sake of simplicity, suppose that $i = 1$ and Gaussian Elimination proceed as follows:

$$\begin{aligned} Q(x) &= ax_1^2 + x_1 f(x_2, \ldots, x_n) + p(x_2, \ldots, x_n) \\ &= a\Big(x_1 + \tfrac{f(x_2,\ldots,x_n)}{2a}\Big)^2 - \tfrac{f(x_2,\ldots,x_n)^2}{4a} + p(x_2, \ldots, x_n) \end{aligned} \tag{2}$$

  where f is a linear combination and p a quadratic form. So, the new quadratic form is:

$$Q'(x) = p(x_2, \ldots, x_n) - \tfrac{f(x_2,\ldots,x_n)^2}{4a} \tag{3}$$

  Example 1. *Let $Q(x) = x_2^2 + x_1x_2 + x_1x_3$. Applying Gaussian Elimination provides two positive squares and one negative.*

$$\begin{aligned} Q(x) &= x_2^2 + x_1x_2 + x_1x_3 \\ &= (x_2 + \tfrac{1}{2}x_1)^2 - \tfrac{1}{4}x_1^2 + x_1x_3 \\ &= (x_2 + \tfrac{1}{2}x_1)^2 - \tfrac{1}{4}(x_1 - 2x_3)^2 + x_3^2 \end{aligned}$$

- If $a_{ii} = 0$, $a_{ij} \neq 0$ and $a_{jj} = 0$ (for some $i, j \in \{1, \ldots, n\}$ and $i \neq j$), eliminate two variables.[3] For sake of simplicity, suppose that $i = 1$ and $j = 2$. So, Gaussian Elimination proceed as follows:

$$\begin{aligned} Q(x) &= ax_1x_2 + x_2 f(x_3, \cdots, x_n) + x_1 g(x_3, \cdots, x_n) + p(x_3, \cdots, x_n) \\ &= (ax_1 + f) \cdot (x_2 + \tfrac{g}{a}) - \tfrac{f \cdot g}{a} + p(x_3, \ldots, x_n) \\ &= \tfrac{1}{4}((l_1 + l_2)^2 - (l_1 - l_2)^2) - \tfrac{f \cdot g}{a} + p(x_3, \ldots, x_n) \end{aligned} \tag{4}$$

  where f, g are linear combinations, p is a quadratic form, $l_1 = (ax_1 + f)$ and $l_2 = (x_2 + \frac{g}{a})$. So, the new quadratic form is:

$$Q'(x) = p(x_3, \ldots, x_n) - \tfrac{1}{a} f(x_3, \ldots, x_n) \cdot g(x_3, \ldots, x_n) \tag{5}$$

  In this case, $Q(x)$ is always indefinite.

  Example 2. *Let $Q(x) = x_1x_2 + x_1x_3 + x_2x_3$. Applying Gaussian Elimination provides one positive square and two negatives.*

$$\begin{aligned} Q(x) &= x_1x_2 + x_1x_3 + x_2x_3 \\ &= (x_1 + x_3)(x_2 + x_3) - x_3^2 \\ &= \tfrac{1}{4}(x_1 + x_2 + 2x_3)^2 - \tfrac{1}{4}(x_1 - x_2)^2 - x_3^2 \end{aligned}$$

[3] In the case of $a_{jj} \neq 0$, it must return to the previous case (Equation (2)).

4. A POLYNOMIAL ALGORITHM

The objective of the algorithms presented in this section is to use Gaussian Elimination to complete the squares and get the precise classification of a given quadratic form. Theorem 3 and Corollary 4 were used to identify positive definite and positive semidefinite quadratic forms, respectively.

Theorem 3. *A quadratic form Q is positive definite if and only if Gaussian Elimination provides n squares affected by the sign "+" [16].*

Corollary 4. *A quadratic form Q is positive semidefinite if and only if Gaussian Elimination provides $k \leq n$ squares and each of them is affected by the sign "+".*

Proof. Gaussian Elimination eliminates one variable and provides a linear form (affected by the sign "+" or "-"), or eliminates two variables and provides two linear forms (one affected by sign "+" and other by "-"). Thus, k must be at most equal n. Let $Q = \sum_{i=1}^{k} c_i l_i^2$, $c_i \neq 0$ and $k \leq n$, obtained by Gaussian Elimination. Since the l_i's are linearly independents, if Q is positive semidefinite, then $c_i \geq 0$ for all i. The other hand follows easily. □

For better organize the subjects, we divided this section in 3 parts: a model of matrix representation to store and update the quadratic form elements; an algorithm that improves the best case of quadratic form classification; and an algorithm that analyses all other cases.

4.1 Model of matrix representation

Algorithms 5 and 6 use a model of matrix representation to store and update the quadratic form elements, during the application of Gaussian Elimination. For this, Q is represented by an upper triangular matrix M_Q, with its entries defined according to the coefficients of each term. Therefore, the coefficient of square terms are placed in the matrix diagonal and the others coefficients are placed above this.

In other words, the coefficient of the term x_1x_2 is placed in M_Q at the row 1 and column 2 as the quotient of two integers. For example, the quadratic form $Q(x) = x_1^2 + 2x_2^2 + 5x_3^2 + 2x_1x_2 - 4x_2x_3$, is represented by an upper triangular matrix as follows:

Example 5. *Matrix representation of $Q(x)$.*

$$M_Q = \begin{bmatrix} \frac{1}{1} & \frac{2}{1} & \frac{0}{1} \\ \frac{0}{1} & \frac{2}{1} & \frac{-4}{1} \\ \frac{0}{1} & \frac{0}{1} & \frac{5}{1} \end{bmatrix}$$

4.2 Best case constant algorithm

This subsection describes an algorithm that explore some situations in which the quadratic form classification can be done in a faster time. Lemmas 6 and 7, and Theorem 8 justify these situations.

Lemma 6. *If $a_{ii} = 0, a_{ij} \neq 0$ and $a_{jj} \neq 0$, for some $i, j \in \{1, \ldots, n\}$, then Q is indefinite.*

Proof. Suppose that $i = 1$ and $j = 2$. We first eliminate x_2. Afterward, the coefficient of x_1^2 is nonzero and we can

eliminate x_1. Follows the Elimination:

$$\begin{aligned}Q(x) &= bx_2^2 + ax_1x_2 + x_2 f(x_3,\cdots,x_n) + x_1 g(x_3,\cdots,x_n) + p(x_3,\cdots,x_n)\\ &= b\left(x_2 + \tfrac{ax_1+f}{2b}\right)^2 - \tfrac{(ax_1+f)^2}{4b} + x_1 g + p\\ &= b\left(x_2 + \tfrac{ax_1+f}{2b}\right)^2 - \tfrac{a^2}{4b}x_1^2 + x_1\left(\tfrac{2bg-af}{2b}\right) - \tfrac{f^2}{4b} + p\\ &= b\left(x_2 + \tfrac{ax_1+f}{2b}\right)^2 - \tfrac{a^2}{4b}\left(x_1 - \tfrac{2bg-af}{a^2}\right)^2 + \tfrac{(2bg-af)^2}{4a^2b} - \tfrac{f^2}{4b} + p\end{aligned}$$

where f and g are linear combinations, p is a quadratic form. Since $sgn(-\frac{a^2}{4b}) = -sgn(b)$, we have that Q is indefinite. □

LEMMA 7. *If $a_{ii} = 0, a_{ij} \neq 0$ and $a_{jj} = 0$, for some $i, j \in \{1,\ldots,n\}$ and $i \neq j$, then Q is indefinite.*

Proof. In this case, Gaussian Elimination will eliminate two variables and provides two linear forms, one affected by sign "+" and other by "-" (see Equation (4)). Therefore, Q is indefinite. □

THEOREM 8. *If Q is positive definite (positive semidefinite) then $a_{ii} > 0$ ($a_{ii} \geq 0$ and if $a_{ii} = 0$ then $a_{ij} = 0$, for all $j \in \{i+1,\ldots,n\}$), for $i \in \{1,\ldots,n\}$.*

Proof. Suppose that $a_{ii} \leq 0$, for some $i \in \{1,\ldots,n\}$. We can begin Gaussian Elimination by a_{ii}. By Theorem 1, Q is not positive definite. If $a_{ii} \neq 0$, then by Corollary 1, Q is not positive definite. Therefore, if $a_{ii} = 0$. Suppose that there exist $j \in \{1,\ldots,n\}$ with $a_{ij} \neq 0$. By Lemas 6 and 7, Q is indefinite and thus cannot be positive semidefinite. □

We use Lemmas 6 and 7, and Theorem 8 to develop a constant algorithm that easily identify some positive definite, semidefinite and indefinite quadratic forms. Algorithm 5 loops through all elements on diagonal of matriz representing the quadratic form and gets the number of positives, negatives and zero elements. Meanwhile, if it finds a negative element and there already is a positive one, or the other way around, the process terminates showing that Q is indefinite.

Otherwise, the number of zero elements are used to identify other indefinite cases. We also developed a polynomial algorithm for the unresolved cases. Algorithm 5 is $O(n^3)$ in the worst case, mainly influenced by the call of QRECOGNITION (see Algorithm 6). However, it is $O(1)$ in the best case. It happens when it finds a negative element and there already is a positive one, or vice versa. And more, the Algorithm 5 is $O(n)$ when the diagonal of M_Q is zero.

4.3 General case polynomial algorithm

Algorithm 6 supposes that Q is not negative definite or semidefinite and tries to identify situations that characterizes definite, semidefinite or indefinite quadratic forms. Observe that it is the case for the first call of Algorithm 6, but not necessarily afterwards. In a simple way, it receives M_Q, complete the first perfect square, update the quadratic form and call itself recursively (doing the same without the line and column of the perfect square found) until M_Q became a simple element or the process fails in some test. The functions in Subsection 3.1 were used to keep the small number of decimal places.

According to Gaussian Elimination, the command at line 8 updates the x_{ii} elements in Q and, the command at line

Algorithm 5: VERIFYDIAG(M_Q)

Input: M_Q = matrix of quadratic form $Q \neq 0$, n = number of coefficients with square terms.
Output: Q classification.

```
1  pos ← 0, neg ← 0, zero ← 0
2  for i ← 1 to n do
3      if a_ii = 0 then
4          zero ← zero + 1
5      else
6          if a_ii > 0 then
7              if neg > 0 then return "Q is
                indefinite!"
8              else pos ← pos + 1
9          else
10             if pos > 0 then return "Q is
                indefinite!"
11             else neg ← neg + 1
12 if zero = n then
13     return "Q is indefinite!"
14 if pos ≠ 0 then          /* pos ≠ 0 and neg ≠ 0 */
15     return QClassification(M_Q, "+", 0)
16 else
17     return QClassification(−M_Q, "-", 0)
```

12 updates the x_{ij} elements, as follows:

$$x_{ii} = a_{ii} - \frac{a_{ii}^2}{4a_{11}}, \quad x_{ij} = a_{ij} - \frac{a_{1i}a_{1j}}{2a_{11}} \tag{6}$$

for $2 \leq i \leq n$ and $i+1 \leq j \leq n$. During this process, if it finds some $a_{ii} < 0$[4], since we check if Q is positive definite, Q should be indefinite.

The variable t is used to identify whether $Q > 0$ or $Q \geq 0$, for positive definite and positive semidefinite cases, and similarly for negative definite and negative semidefinite cases. The loop command at line 3 identifies semidefinite and indefinite cases, according to Lemmas 6 and 7.

Since the loop commands in lines 7 and 11 (used to make Gaussian Elimination) has cost $O(n^2)$ and it may call itself recursively $n-1$ times, the Algorithm 6 is $O(n^3)$ in the worst case (the complexity of *gcd* function has been lessened throughout the whole solution).

THEOREM 9. *Algorithm 6 is correct.*

Proof. We first observe that every call of Algorithm 6 decreases the number of lines and columns of M_Q in 1 (see line 13). Thus, eventually $n = 1$, the test in line 1 is *False* and the algorithm terminates. Otherwise, the process fails on the test at line 4 or on the test at line 9 and the algorithm terminates.

Recall that Q is negative definite (negative semidefinite) if and only if $-Q$ is positive definite (positive semidefinite). The fact if we deal with Q or $-Q$ is defined in the parameter t as "+" or "-". Observe that, in Algorithm 5, if $pos \neq 0$ ($neg \neq 0$) then Q ($-Q$) cannot be negative definite nor negative semidefinite. Thus, the algorithm just verify if Q ($-Q$) is positive definite, positive semidefinite or indefinite.

We prove the correctness of QRECOGNITION by induction on n. The base case is when $n = 1$. In this case, M_Q has one

[4]At beginning, $a_{ii} \geq 0$ for all i.

Algorithm 6: QCLASSIFICATION($M_Q, t, szero$)

Input: M_Q = matrix of quadratic form Q, t ="+" or "-".

Output: Q classification.

```
1  if n ≥ 2 then
2      if a_11 = 0 then
3          for k ← 2...n do
4              if a_1k ≠ 0 then
5                  return "Q is indefinite!"
6          szero ← 1
7      else
8          for i ← 2...n do
9              a_ii ← SUB(a_ii, DIV(MULT(a_1i, a_1i),
                 MULT([4, 1], a_11)))
10             if a_ii < 0 then
11                 return "Q is indefinite!"
12             for j ← i + 1...n do
13                 a_ij ← SUB(a_ij, DIV(MULT(a_1i, a_1j),
                     MULT([2, 1], a_11)))
14     QCLASSIFICATION(M_Q(2...n, 2...n), t, szero)
15 else
16     if a_11 = 0 then szero ← 1
17     if szero = 0 then
18         if t = "+" then
19             return "Q is positive definite!"
20         else
21             return "Q is negative definite!"
22     else
23         if t = "+" then
24             return "Q is positive semidefinite!"
25         else
26             return "Q is negative semidefinite!"
```

element $a_{11} \geq 0$. If $a_{11} \neq 0$, then Q is positive definite. If not, Q is positive semidefinite. Let M_Q be an $n \times n$ matrix. At line 2 to 4, the algorithm verify the condition of Lemmas 6 and 7. If it is valid, we have that Q is indefinite.

If $a_{11} = 0$, in the line 5, then $zero = 1$. In this case, $a_{1j} = 0$ for all j and x_1 does not appears in Q. In the line 13, we use the algorithm to see the classification of $Q' = Q$, where M'_Q is $(n-1) \times (n-1)$ matrix. Since $szero = 1$, we can have that Q in indefinite or positive semidefinite. If $a_{11} \neq 0$, the lines 7 to 12 calculate Q' according to Gaussian Elimination. Thus, $Q = a_{11}l^2 + Q'$ for some linear combination l. At line 9, we verify the new $a_{ii} \geq 0$ for all i, since if not we will have that Q' is not positive definite nor positive semidefinite. Thus, Q is indefinite.

Therefore, when called QRECOGNITION at line 13 for Q', all elements of the diagonal of M'_Q are non-negative. Moreover, Q is positive definite (positive semidefinite) if and only if Q' is positive definite (positive semidefinite). □

THEOREM 10. *Algorithm 5 is correct.*

Proof. Algorithm 5 uses the command at line 2 to analyses the n elements on M_Q diagonal. And if necessary, call the Algorithm 6 for unresolved cases. Therefore, since Algorithm 6 terminates (see Theorem 9), Algorithm 5 terminates. To prove that Algorithm 5 shows the correct result, we have to analyse three different situations according to Gaussian Elimination:

1. $zero = n$, Q is indefinite (line 13). In this case, $a_{ii} = 0$ for all $i \in \{1, \ldots, n\}$. Assuming that Q is a nonzero quadratic form, the method eliminates two variables, provides two squares and Q is indefinite (see Lemma 7).

2. $a_{ii} > 0$ and $neg \neq 0$, Q is indefinite (line 7). This conditions means that the process find an $a_{ii} > 0$ and there already is at most one $a_{jj} < 0$, for $j \in \{1, \ldots, i-1\}$, in Q. We can begin Gaussian Elimination by a_{ii} or a_{jj}. By Theorem 1 and Corollary 1, we conclude that Q must be indefinite.

3. $a_{ii} < 0$ and $pos \neq 0$, Q is indefinite (line 10). It follows easily as in 2.

Since the Algorithm 5 shows the correct result and terminates, it is correct. □

5. FINAL REMARKS AND COMMENTS

The algorithms presented in this paper have a polynomial cost ($O(1)$ in the best case and $O(n^3)$ in the worst case) and are similar to the best known results (which are iterative and expansive) to identify the type of a quadratic form. In this work, an important issue with respect to the algorithms is the considerable improvement in the accuracy of the results.

In order to make a comparison, our algorithms present the exact type of the quadratic form (positive definite, positive semidefinite, etc.). Some strategies such as Cholesky Decomposition presents a number, probably with a large number of decimal places, which may have been truncated to facilitate processing and, therefore, harm the analysis of the results. For example, a positive definite quadratic form can be interpreted as positive semidefinite if the result 0.0000000001 is truncated to 0.000.

Gaussian Elimination to complete the squares allowed a direct look at a simple aspect, but very efficient. The sign that affects every perfect square is a reliable way to identify linear forms that are linearly independent and, using simple strategies to complete the squares, we could determine the type of a quadratic form.

The strategy of representing rational numbers as a fraction of integers use basic and direct operations that maintain a reliable result (with a small number of decimal places). This process is extremely important because it ensure an accurate result that is easy to be identified (unlike the results provided by iterative algorithms).

6. ACKNOWLEDGMENTS

The authors would like to thank Instituto Federal Goiano Campus - Morrinhos and Universidade Federal de Goiás for the support during this research.

7. REFERENCES

[1] N. Alon, K. Makarychev, Y. Makarychev, and A. Naor. Quadratic forms on graphs. *Inventiones mathematicae*, 163(3):499–522, 2006.

[2] J. Alves, D. Castongay, and T. Brüstle. A polynomial recognition of unit forms using graph-based strategies. *Discrete Applied Mathematics*, 253:61–72, 2019.

[3] J. Alves, D. Castonguay, and T. Brüstle. Unit form recognition by mutations: Application of mutations in the search of positive roots. *Discrete Applied Mathematics*, 291:223–236, 2021.

[4] E. Bayer-Fluckiger, D. Lewis, and A. Ranicki. Quadratic forms and their applications. In *Proceedings of the Conference on Quadratic Forms and Their Applications (University College Dublin*, 1999.

[5] M. Berger. *Geometry I*. Springer, 2009.

[6] R. L. Burden and J. D. Faires. *Numerical analysis 8th ed*. 2005.

[7] R. Byers. A hamiltonian-jacobi algorithm. *IEEE transactions on automatic control*, 35(5):566–570, 1990.

[8] A. Cosentino and S. Severini. Weight of quadratic forms and graph states. *Physical Review A*, 80(5):052309, 2009.

[9] Z. Drmač and K. Veselić. New fast and accurate jacobi svd algorithm. i. *SIAM Journal on matrix analysis and applications*, 29(4):1322–1342, 2008.

[10] R. M. Freund. Quadratic functions, optimization, and quadratic forms. *Massachusetts Institute of Technology*, 2004.

[11] L. J. Gerstein. Integral quadratic forms and graphs. *Linear Algebra and its Applications*, 585:60–70, 2020.

[12] L. Giraud, J. Langou, and M. Rozloznik. The loss of orthogonality in the gram-schmidt orthogonalization process. *Computers & Mathematics with Applications*, 50(7):1069–1075, 2005.

[13] L. Giraud, J. Langou, M. Rozložník, and J. van den Eshof. Rounding error analysis of the classical gram-schmidt orthogonalization process. *Numerische Mathematik*, 101(1):87–100, 2005.

[14] G. Helmberg, P. Wagner, and G. Veltkamp. On faddeev-leverrier's method for the computation of the characteristic polynomial of a matrix and of eigenvectors. *Linear algebra and its applications*, 185:219–233, 1993.

[15] N. J. Higham. Cholesky factorization. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(2):251–254, 2009.

[16] N. J. Higham. Gaussian elimination. *Wiley Interdisciplinary Reviews: Computational Statistics*, 3:230–238, 2011.

[17] A. Lerario. *Convex pencils of real quadratic forms*, volume 48. Springer, 2012.

[18] O. T. O'Meara. *Introduction to quadratic forms*, volume 117. Springer, 2013.

[19] C. C. Paige, M. Rozloznik, and Z. Strakos. Modified gram-schmidt (mgs), least squares, and backward stability of mgs-gmres. *SIAM Journal on Matrix Analysis and Applications*, 28(1):264–284, 2006.

[20] R. Rehman and I. C. Ipsen. Computing characteristic polynomials from eigenvalues. *SIAM journal on matrix analysis and applications*, 32(1):90–114, 2011.

[21] J. Tattersall. Finding the greatest common divisor. *Mathematical Time Capsules: Historical Modules for the Mathematics Classroom*, (77):199, 2011.

[22] H. A. van der Vorst. Computational methods for large eigenvalue problems. 2002.