CLOSED-FORM TEST FOR INTERSECTIONS OF PARAMETRIC CUBICS.

PRZEMYSŁAW KOPROWSKI

Abstract. We describe a closed-form test detecting whether two cubic segments intersect. The presented algorithms is purely algebraic and so is not restricted to any particular representation of cubic segments. One of main advantages of the proposed algorithm is that it gives a definite answer even for "difficult" intersections like: tangent intersections, intersections at singularities or intersections of segments that are not relatively prime.

1. Introduction and related works

Following [8], in this whole paper we use the term *cubic curve* when we refer to an unbounded curve (of infinite length) of degree three, while by a *cubic segment* we understand a subset of a curve corresponding to the unit interval $[0, 1]$ in some parametrization. The problem of detecting whether two given cubic segments intersect each other is of vital importance for computer graphics and CAGD. For example, if two segments of a spline, representing a cross-section of a surface, intersect, then such a surface cannot be manufactured.

This paper was inspired by the discussion hold on `comp.graphics.algorithms` Usenet group in September 2004. A general believe expressed in the discussion was that it is not possible to construct a closed-form test for polynomial Bézier cubics. Contrary to this believe, we show that such a test in fact exists. The algorithm is not genuinely new, it is just an application of a general method known in computational real algebraic geometry, to the specified task. Unfortunately, the algorithm, albeit closed-form, is unbearably slow and as such not fully suitably for real-life applications (at least using the current hardware technology). Consequently, the paper serves a double purpose: one aim is to disprove a false general believe. The other one, and of equal importance, is to propose some optimizations to a generic algorithms taking into account the peculiarities of the task at hand. The optimization (giving roughly 50% time gain) are discussed in Section 3 of this paper. Moreover, the fact that there exists at least one such a closed-form solution may encourage research for other closed-form algorithms.

The bibliography on the subject of intersections of parametric cubics is rich. Here we present only a very brief listing of related works. For a more extended list we refer the reader to the references in cited papers. A basic approach to investigate intersections of two cubic segments represented in Bézier-Bernstein basis is to use the convex hull property, together with a subdivision. Hence, a basic algorithm boils down to: check if the convex hulls of the two segments intersect. If not, then the segments cannot intersect. Otherwise, split each segment in half and repeat the procedure. Stop once the convex hulls' diameters are smaller than a given tolerance. This is a recursive, numerical algorithm which may easily return false results (if the tolerance is incorrectly chosen), since it uses only a rejection test (i.e. a necessary condition which is clearly not sufficient). Such a subdivision algorithm

Date: November 19, 2012.
Key words and phrases. curve intersections, Gröbner bases, zero-dimensional systems.

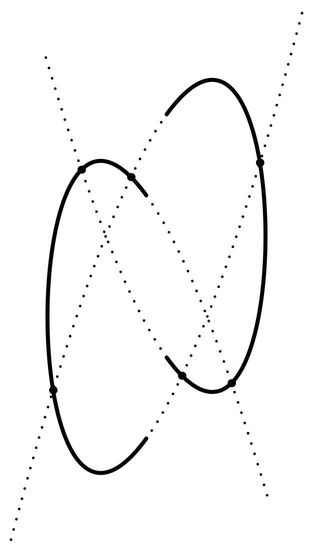


FIGURE 1. Two segments with six apparent intersections.

can be easily augmented with a sufficient condition (which in turn is not necessary), using the fact that if a control polygon of a Bézier segment crosses a given line an odd number of times, then the Bézier segment itself crosses the same line at least once. This way one obtains an algorithm which gives a definite answer for traversal intersections. Recently [11] presented a variation of a subdivision algorithm that uses a necessary and sufficient condition for intersections of so called *elementary Bézier segments*. This is a promising approach, unfortunately it does not cover all the possible cases. Usually people are interested not only in testing whether two segments intersect or not, but in finding the actual intersection points. This problem is discussed in details in [4, 5]. Another classical approach to test for intersections of two parametric *curves* (or a *curve* and a *segment*) is to convert a curve to its implicit form (for details see e.g. [9]) and substitute the parametric equation of the other curve/segment into the implicit equation of the former one. Now, Sturm's theorem may be used to verify whether the obtained equation has solutions in the interval of our interest. Although this method is closed-form by its nature (the length of the Sturm sequence is bounded by nine in our case and so the total number of operations is bounded by a constant, too) it cannot be, unfortunately, used to test whether two *segments* intersect or not. To understand why, consider the two cubics depicted in Figure 1. Each segment intersects the other curve at three distinct points, nevertheless, the two segments are clearly disjoint and so a test based on Sturm's theorem would give false positives here.

2. The basic algorithm

Let $C_0(t) = \big(x_0(t), y_0(t)\big)$ and $C_1(s) = \big(x_1(s), y_1(s)\big)$ be two plane polynomial cubic segments with t, s varying over $[0,1]$, each. The two segments intersect if and only if the system

$$\begin{cases} x_0(t) - x_1(s) = 0 \\ y_0(t) - y_1(s) = 0 \end{cases} \tag{1}$$

has a solution in the unit square $[0,1]^2$. Denote $f(t,s) := x_0(t) - x_1(s)$ and $g(t,s) := y_0(t) - y_1(s)$. Our first step is to determine whether Eq. (1) is a zero-dimensional system. To this end compute a Gröbner basis $\mathcal{G}$ of the ideal $\langle f, g\rangle \lhd \mathbb{R}[t,s]$ and then check if there is $N \in \mathbb{N}$ such that s^N and t^N belong to the ideal $\mathcal{J}$ generated by the leading monomials of $\mathcal{G}$. Now, [1, Theorem 2.2.7] asserts that Eq. (1) is zero-dimensional if and only if such an N exists.

2.1. **Zero-dimensional case.** If the system Eq. (1) is zero-dimensional, then $\mathcal{A} := \mathbb{R}[t,s]/\langle f,g\rangle$ is a finitely-dimensional $\mathbb{R}$-algebra with a basis $\mathscr{B} = \{s^m t^n \notin J\}$ by [1, Theorem 2.2.7]. The dimension of $\mathcal{A}$ is not greater than 9 since f, g are cubic with respect to both variables. Let L_h denote an endomorphism of $\mathcal{A}$ defined as $L_h(x) := h \cdot x$ and let

$$\operatorname{Her}(h) := \Big[\operatorname{Tr}\big(L_{h\alpha\beta}\big)\Big]_{\alpha,\beta\in\mathscr{B}}$$

be the Hermite matrix of h. It is symmetric, hence diagonalizable over the reals. Denote by $\mathrm{SQ}(h)$ the signature of the associated (symmetric) bilinear form $(x,y) \mapsto x^T \cdot \operatorname{Her}(h) \cdot y$.

Consider two polynomials $q(t) = (1-t)t$ and $r(s) = (1-s)s$. The unit square $[0,1]^2$ is precisely the set where they are simultaneously non-positive. For any $i, j \in \{-1,0,1\}$ denote

$$c_{i,j} = \sharp\big\{(t,s) \in \mathbb{R}^2 : f(t,s) = g(t,s) = 0 \text{ and } \operatorname{sgn} q(t) = i, \operatorname{sgn} r(s) = j\big\}$$

the number of solutions of Eq. (1) with prescribed signs of q and r. It follows that the number of solutions of Eq. (1) in the unit square equals $\Delta = c_{-1,-1} + c_{-1,0} + c_{0,-1} + c_{0,0}$. Compute $\mathrm{SQ}(q^k r^l)$ for every $k, l \in \{0,1,2\}$. Pedersen-Roy-Spirglas theorem [2, Theorem 4.2.7] asserts that the following equations hold

$$\begin{cases} c_{-1-1} + c_{-10} + c_{-11} + c_{0-1} + c_{00} + c_{01} + c_{1-1} + c_{10} + c_{11} & = \mathrm{SQ}(1) \\ -c_{-1-1} - c_{-10} - c_{-11} + c_{1-1} + c_{10} + c_{11} & = \mathrm{SQ}(q) \\ c_{-1-1} + c_{-10} + c_{-11} + c_{1-1} + c_{10} + c_{11} & = \mathrm{SQ}(q^2) \\ -c_{-1-1} + c_{-11} - c_{0-1} + c_{01} - c_{1-1} + c_{11} & = \mathrm{SQ}(r) \\ c_{-1-1} + c_{-11} + c_{0-1} + c_{01} + c_{1-1} + c_{11} & = \mathrm{SQ}(r^2) \\ c_{-1-1} - c_{-11} - c_{1-1} + c_{11} & = \mathrm{SQ}(qr) \\ -c_{-1-1} + c_{-11} - c_{1-1} + c_{11} & = \mathrm{SQ}(q^2 r) \\ -c_{-1-1} - c_{-11} + c_{1-1} + c_{11} & = \mathrm{SQ}(qr^2) \\ c_{-1-1} + c_{-11} + c_{1-1} + c_{11} & = \mathrm{SQ}(q^2 r^2). \end{cases} \tag{2}$$

Solve the system to get

$$\Delta = \mathrm{SQ}(1) - \frac{1}{2}\big(\mathrm{SQ}(q) + \mathrm{SQ}(q^2) + \mathrm{SQ}(r) + \mathrm{SQ}(r^2)\big) + \\ + \frac{1}{4}\big(\mathrm{SQ}(qr) + \mathrm{SQ}(q^2 r) + \mathrm{SQ}(qr^2) + \mathrm{SQ}(q^2 r^2)\big).$$

Clearly, $\Delta \neq 0$ if and only if the two segments intersect. The number of steps to verify this condition is clearly bounded by a constant: there are at most nine Hermite matrices to compute and their dimensions do not exceed nine. Thus the technique is closed-form, albeit slow.

2.2. **Non-zero-dimensional case.** Assume that the system Eq. (1) is not zero-dimensional. This may only happen when C_0 and C_1 are two segments of a single cubic curve. If this is the case, they may have 0, 1, 2 or infinitely many intersection points. The key is to express the endpoints of one segment using the parametrization of the other one. Let B be the Bézoutian of the polynomials $x - x_0(t)$ and $y - y_0(t)$ with respect to the variable t. Then the determinant $\det B$ is the resultant $\operatorname{Res}_t\big(x - x_0(t), y - y_0(t)\big)$ and the implicit form of the curve containing both segments in question is given by the equation

$$\det B = 0.$$

Use the inversion formula (see e.g. [6]) to find the parameters $\alpha < \beta$ corresponding to the two endpoints of C_1. There are six cases that we must consider:

(I) if $\alpha \in [0,1)$ or $\beta \in (0,1]$ then the two segments overlap and so they have infinitely many intersection points;

(II) if $\alpha < 0$ and $\beta > 1$ then C_0 is a sub-segment of C_1 and so they again have infinitely many intersections;

(III) if C is not a crunodal curve, or is crunodal but the crunode does not belong to at least one of C_0, C_1, and either $\alpha = 1$ or $\beta = 0$ then C_0, C_1 have exactly one intersection point being their common endpoint;

(IV) if C is crunodal and the crunode belongs to both C_0 and C_1 then:
   (a) if either $\alpha = 1$ or $\beta = 0$ then there are two distinct intersection points (the crunode and the common endpoint);
   (b) if $[\alpha, \beta] \cap [0,1] = \emptyset$ then the crunode is the only intersection point;

(V) in every other case the segments C_0, C_1 do not intersect.

Observe that cases III–IV require a (closed-form) procedure checking whether a given segment is crunodal. An efficient test was presented in [10]. In order to check if the two segments simultaneously contain a crunode let $a < b \leq c < d$ be the parameters 0, 1, α, β of the endpoints of C_0 and C_1 sorted in the ascending order. If we have already ruled out the case of overlapping segments (points I and II above), then a, b correspond to one segment and c, d to the other one. Thus:

Observation 3. *The crunode belongs to both segments if and only if the segment parametrized by the interval $[a,d]$ is crunodal while the arcs corresponding to $[a,c]$ and $[b,d]$ are not.*

3. Optimizations of the basic algorithm

The intersection test presented in the previous section is closed-form. The number of operations is bounded by a constant. Unfortunately this constant is huge, rendering the method quite impractical. Here we propose few possible optimizations. The main source of inefficiency is the fact that for every Hermite matrix we need to compute d^2 traces, where d is the dimension of $\mathcal{A}$ and the number of complex intersection point (counting with multiplicities) at the same time by [2, Lemma 4.58] and [3, Corollary IV.2.5]. Thus $d \leq 9$ by Bézout theorem and this bound can be reached (see the fourth example in the last section). If all the intersections are traversal, then the simple recursive algorithm described in the introduction is most effective. Thus it is natural to couple these two algorithms and use the one described in this paper only when the iterative method fails to give a definitive answer, combining the best of the two world. This means that the described algorithm will be used only for "difficult case" when there occurs intersections of higher multiplicities. Fortunately this is the case, where serious optimizations are possible. First of all we may reduce the dimension of the algebra taking the radical $\sqrt{\langle f, g\rangle}$ instead of the ideal $\langle f, g\rangle$ itself.

Proposition 4. *Under the above assumptions, let $r_t := \operatorname{Res}_s(f,g)$, $r_s := \operatorname{Res}_t(f,g)$ and $\overline{r}_t := {}^{r_t}\!/_{\gcd(r_t, r_t')}$, $\overline{r}_s := {}^{r_s}\!/_{\gcd(r_s, r_s')}$. Then $\sqrt{\langle f, g\rangle} = \langle f, g, \overline{r}_t, \overline{r}_s\rangle$.*

Proof. This follows immediately from [3, Proposition II.2.7] and standard properties of the resultant. □

Corollary 5. *If there are intersections of higher multiplicities, then*

$$\dim \mathbb{R}[t,s]/\langle f, g, \overline{r}_t, \overline{r}_s\rangle < \dim \mathbb{R}[t,s]/\langle f, g\rangle,$$

while both algebras are associated to the same set of intersection points.

One additional advantage of computing the resultants is that we can earlier decide whether the system is zero-dimensional or not, since in the later case the resultant is null. This way we avoid an expensive step of computing a Gröbner basis, when it is not really necessary. In the main body of the algorithm, we compute nine Hermite matrices $\mathrm{Her}(q^i r^j)$ for $0 \le i, j \le 2$. This is the most time-consuming part of the procedure. We may significantly reduce the computation time building only four Hermite matrices $\mathrm{Her}(q^i r^j)$ for $1 \le i, j \le 2$. Indeed, we have $\Delta = c_{-1,-1} + c_{-1,0} + c_{0,-1} + c_{0,0}$. The last four rows of Eq. (2) let us compute the first term

$$c_{-1,-1} = \frac{1}{4}\bigl(\mathrm{SQ}(qr) + \mathrm{SQ}(q^2 r) + \mathrm{SQ}(qr^2) + sq(q^2 r^2)\bigr).$$

The three remaining terms may be computed with no references to Pedersen-Roy-Spirglas theorem. By the definition, $c_{0,0}$ is non-zero if and only if the two segments share a common endpoint, which is straightforward to check. If this is the case, the algorithm may return TRUE without any further computations. Similarly, $c_{-1,0}$ (respectively $c_{0,-1}$) is non-zero if and only if the first (resp. second) segment passes through any of the endpoints of the other segment. This condition may be easily checked expressing the segment in an implicit form $r = 0$, with $r = \mathrm{Res}_t\bigl(x - x_0(t), y - y_0(t)\bigr)$. If either $r\bigl(x_1(0), y_1(0)\bigr) = 0$ or $r\bigl(\bigl(x_1(1), y_1(1)\bigr) = 0$, then use the inversion formula to recover the corresponding parameter t. If it belongs to $(0, 1)$, then the algorithm may again terminate returning TRUE. Analogously one verifies whether $c_{0,-1} \ne 0$.

We may now present the pseudo-code of the optimized algorithm. The initialization part is presented in Algorithm 1. Next, Algorithm 2 shows the first sub-procedure and Algorithm 3 the other sub-procedure.

4. Example results

In order to verify the correctness of the method as well as the effectiveness of the proposed optimizations, the algorithm was implemented in a computer algebra system Sage [7]. All tests were run on 9 different computers. The first test suite consisted of 14 specifically tailored pairs of curves with "difficult" intersections. Table 1 shows how the optimized algorithm behaved, when compared to the generic one. For the second test suite, each computer generated 200 random pairs of cubic Bézier curves ans tested for occurrences intersections using both versions of the algorithm. In this test, the average running time of the optimized algorithm was better by « **TODO** »when compared with the generic one. Concluding, the optimized algorithm proposed in this paper is about twice faster than the generic one due to the fact that it requires only four Hermite matrices and it reduces the dimension of the algebra $\mathcal{A}$ if there are intersections of higher multiplicities present.

References

[1] W. W. Adams and P. Loustaunau. *An introduction to Gröbner bases*, volume 3 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 1994.

[2] S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in real algebraic geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer-Verlag, Berlin, 2003.

[3] D. Cox, J. Little, and D. O'Shea. *Using algebraic geometry*, volume 185 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1998.

[4] D. Manocha and J. W. Demmel. Algorithms for intersecting parametric and algebraic curves I: Simple intersections. *ACM Trans. Graph.*, 13(1):73–100, 1994.

[5] D. Manocha and J. W. Demmel. Algorithms for intersecting parametric and algebraic curves II: Multiple intersections. *Graph. Models Image Process.*, 57(2):81–100, 1995.

[6] T. W. Sederberg. Applications to computer aided geometric design. In *Applications of computational algebraic geometry (San Diego, CA, 1997)*, volume 53 of *Proc. Sympos. Appl. Math.*, pages 67–89. Amer. Math. Soc., Providence, RI, 1998.
[7] W. Stein et al. *Sage Mathematics Software (Versions: 4.7.1 and 5.3).* The Sage Development Team, 2010–12. `http://www.sagemath.org`.
[8] M. Stone and T. DeRose. A geometric characterization of parametric cubic curves. *ACM Trans. Graph.*, 8(3):147–163, 1989.
[9] B. Sturmfels. Introduction to resultants. In *Applications of computational algebraic geometry (San Diego, CA, 1997)*, volume 53 of *Proc. Sympos. Appl. Math.*, pages 25–39. Amer. Math. Soc., Providence, RI, 1998.
[10] S. Vincent. Fast detection of the geometric form of two-dimensional cubic Bézier curves. *Journal of Graphics Tools: JGT*, 7(3):43–51, 2002.
[11] C. K. Yap. Complete subdivision algorithms, i: intersection of bezier curves. In *Proceedings of the twenty-second annual symposium on Computational geometry*, SCG '06, pages 217–226, New York, NY, USA, 2006. ACM.

E-mail address: `pkoprowski@member.ams.org`

Instytut Matematyki
Uniwersytet Śląski
ul. Bankowa 14
PL-40-007 Katowice, Poland

```
**Input**: $\begin{cases} C_0(t) = \big(x_0(t), y_0(t)\big) & \text{first cubic segment} \\ C_1(s) = \big(x_1(s), y_1(s)\big) & \text{second cubic segment} \end{cases}$
**Output**: $\begin{cases} \textbf{true} & \text{if the two segments intersect} \\ \textbf{false} & \text{otherwise} \end{cases}$
$f_0(t,s) \leftarrow x_0(t) - x_1(s)$;
$f_1(t,s) \leftarrow y_0(t) - y_1(s)$;
$r_t \leftarrow \mathrm{Res}_s(f_0, f_1)$;
$r_s \leftarrow \mathrm{Res}_t(f_0, f_1)$;
**if** $r_t = 0$ **then**
    **return** NonZeroDimensionalIntersections$(C_0, C_1)$
$\overline{r}_t \leftarrow r_t / \gcd(r_t, r_t')$;
$\overline{r}_s \leftarrow r_s / \gcd(r_s, r_s')$;
$\mathcal{G} \leftarrow$ GroebnerBasis$(\{f_0, f_1, \overline{r}_t, \overline{r}_s\})$;
$\mathcal{J} \leftarrow$ InitialMonomials$(\mathcal{G})$;
**return** ZeroDimensionalIntersections$(C_0, C_1, \mathcal{G}, \mathcal{J})$;
```
Algorithm 1: Initialization part of the algorithm

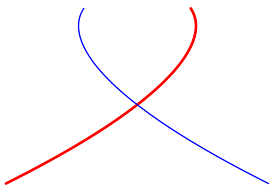
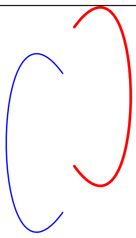
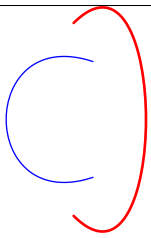
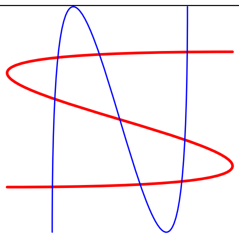
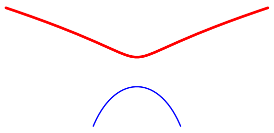
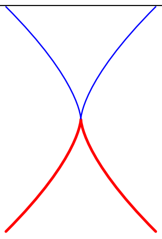
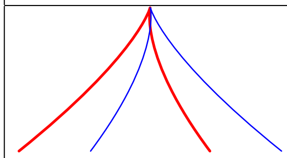
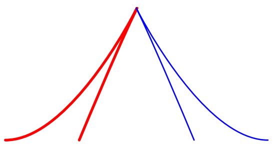
```
Input: { $C_0(t) = (x_0(t), y_0(t))$   first cubic segment
       { $C_1(s) = (x_1(s), y_1(s))$   second cubic segments
       { $\mathcal{G}$                  Gröbner basis
       { $\mathcal{J}$                  the initial monomials
Output: { true   if the two segments intersect
        { false  otherwise
if $C_0(0) = C_1(0)$ or $C_0(0) = C_1(1)$ or $C_0(1) = C_1(0)$ or $C_0(1) = C_1(1)$ then
    return true;
$B_0 \leftarrow$ BezoutMatrix$(x - x_0(t), y - y_0(t))$;
$r_0 \leftarrow \det B_0$;
if $r_0(C_1(0)) = 0$ then
    if InversionFormula$(B, C_1(0)) \in (0, 1)$ then
        return true;
if $r_0(C_1(1)) = 0$ then
    if InversionFormula$(B, C_1(1)) \in (0, 1)$ then
        return true;
$B_1 \leftarrow$ BezoutMatrix$(x - x_1(t), y - y_1(t))$;
$r_1 \leftarrow \det B_1$;
if $r_1(C_0(0)) = 0$ then
    if InversionFormula$(B, C_0(0)) \in (0, 1)$ then
        return true;
if $r_1(C_0(1)) = 0$ then
    if InversionFormula$(B, C_0(1)) \in (0, 1)$ then
        return true;
$S \leftarrow \{s^m t^n : s^m t^n \notin \mathcal{J}, m, n \in \mathbb{N}_0\}$;
$d \leftarrow \sharp S$;
$q \leftarrow (1 - t)t$;
$r \leftarrow (1 - s)s$;
$V \leftarrow \{qr, q^2 r, qr^2, q^2 r^2\}$;
for $f \in V$ do
    $\overline{f} \leftarrow$ NormalForm$(f, \mathcal{G})$;
    for $0 \leq i, j < d$ do
        $\mathrm{Her}[i, j] \leftarrow \mathrm{Tr}(L_{\overline{f} \cdot S[i] \cdot S[j]})$;
    $SQ[f] \leftarrow$ Signature$(\mathrm{Her})$;
if $\sum_{f \in V} SQ[f] \neq 0$ then
    return true;
else
    return false;
```

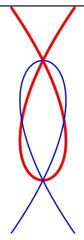
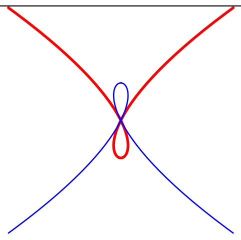
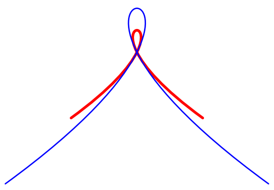
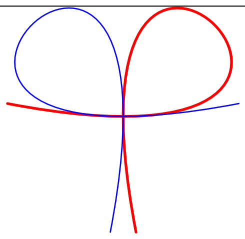
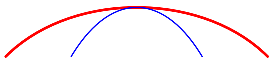
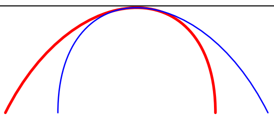
Algorithm 2: The zero-dimensional case

```
Input: $\begin{cases} C_0(t) = \big(x_0(t), y_0(t)\big) \\ C_1(s) = \big(x_1(s), y_1(s)\big) \end{cases}$ two segments of the same cubic curve
Output: $\begin{cases} \textbf{true} & \text{if the two segments intersect} \\ \textbf{false} & \text{otherwise} \end{cases}$
$B \leftarrow$ BezoutMatrix$\big(x - x_0(t), y - y_0(t)\big)$;
$t_0 \leftarrow$ InversionFormula$\big(B, C_1(0)\big)$;
$t_1 \leftarrow$ InversionFormula$\big(B, C_1(1)\big)$;
if $t_0 < t_1$ then
    $\alpha \leftarrow t_0$; $\beta \leftarrow t_1$;
else
    $\alpha \leftarrow t_1$; $\beta \leftarrow t_0$;
if $(0 \le \alpha \le 1)$ or $(0 \le \beta \le 1)$ then
    return true;
if $(\alpha < 0)$ and $(\beta > 1)$ then
    return true;
if $\alpha < 0$ then
    $a \leftarrow \alpha$; $b \leftarrow \beta$; $c \leftarrow 0$; $d \leftarrow 1$;
else
    $a \leftarrow 0$; $b \leftarrow 1$; $c \leftarrow \alpha$; $d \leftarrow \beta$;
$N_{ad} \leftarrow$ IsNodal$(C_0 \mid_{[a,d]})$;
$N_{ac} \leftarrow$ IsNodal$(C_0 \mid_{[a,c]})$;
$N_{bd} \leftarrow$ IsNodal$(C_0 \mid_{[b,d]})$;
if $N_{ad}$ and not $N_{ac}$ and not $N_{bd}$ then
    return true
else
    return false
```

Algorithm 3: The non-zero-dimensional case

TABLE 1. Time comparison of generic and optimized algorithms.

segments' arrangement	running time of the optimized algorithm vs. the generic one	segments' arrangement	running time of the optimized algorithm vs. the generic one
 Non-zero-dimensional intersection	98.71%	 Six apparent intersections	49.83%
 Six apparent intersection	50.54%	 Nine simple intersections	48.57%
 Acnodal apparent intersection	50.06%	 Cusp-cusp intersection	51.77%
 Cusp-cusp intersection	48.99%	 Cusp-cusp intersection	48.20%

 Nodal intersection	50.15%	 Nodal intersection	47.54%
 Nodal intersection	47.59%	 Nodal intersection	50.43%
 Tangent non-crossing intersection	47.84%	 Tangent crossing intersection	48.07%