On Root Classification in Kinetic Data Structures

Tomáš Vomáčka, Ivana Kolingerová
Faculty of Applied Sciences
Univerzity of West Bohemia
Pilsen, Czech Republic
tvomacka@kiv.zcu.cz; kolinger@kiv.zcu.cz

***Abstract*—In this paper we discuss the mathematical properties of kinetic events computation for kinetic data structures with polynomial-type certificate functions. We show that it is neither theoretically possible nor numerically safe to ignore the multiplicities of roots of these equations. The multiplicities of the roots are sometimes ignored in order to speed up the process of estimating their location, however, they must be taken into account during the management of the kinetic data structures. Some of the roots obtained by the computations of these equations do not necessarily carry the expected information (i.e., the times of future kinetic events) and they may be therefore avoided entirely during the computation. This text shows how to distinguish these roots before their exact location is computed and thus to avoid their computation.**

***Keywords*-Computational geometry, Polynomials, Data structures.**

I. Introduction

Since many modern applications use a set of moving primitives (points, triangles) or even more complex objects as the input data, the kinetic data structures (KDS). Note that this term is used in the meaning of a set of rules that defines the mutual relationships for a given set of primitives – the eventual implementation is not considered. KDS represent a valid tool in applications such as collision detection, physical and mathematical simulations, simulations of crowds, etc. We may also encounter them in less obvious applications such as kinetic-based management of function envelopes or motion interpolation.

The mathematical properties of KDS are rarely discussed in literature. In this paper, we are going to explore one specific mathematical property of kinetic data structures. Since the general topology of each kinetic data structure has to change as a result of the movement of the input data, we need to compute the time instants of these changes. Experiments have shown that this particular task is the most time-consuming part of KDS management. Therefore it would be very convenient to be able to either speed up the computation or omit some of the calculations. We show that it is possible to ignore a nontrivial part of the computation based solely on the algebraic properties of the given equations.

This text will be organized in the following fashion: in Section II, the previous work in the field of kinetic data structures will be discussed. Section III will focus on the basic principles and properties of the kinetic data structures. Section IV will discuss the types of roots one may encounter during the management of kinetic data structures. Section V will summarize various aspects of our research and Section VI will provide the reader with the results of our work and will conclude the paper.

II. State of the Art

A. Kinetic Data Structures Use

Kinetic data structures were first introduced by Basch et al. in [1] as a means of maintaining several different data structures (such as 2D convex hull or an envelope of convex functions) over a set of moving points with the aim to create an apparatus for managing a kinetic Voronoi diagram. Since then, the kinetic data structures have been undergoing an extensive research (with a special focus on the spatial division kinetic data structures such as the aforementioned Voronoi diagram and Delaunay triangulation) – see [2]–[6] and others. There are many different fields of application of KDS: kinetic Delaunay triangulation was proposed as a means of detecting collisions in [4]. Examples of this approach include the collision detection between convex polygons by Erickson et al. who showed in [7] that the kinetic approach may be used to detect collisions between convex polygons in E^2. Their work was further extended by Guibas et al. in [8] by employing a kinetic regular triangulation for managing the bounding spheres of the moving polyhedra. Practical use may include such applications as the one proposed by Goralski and Gold in [9] which uses the kinetic Voronoi diagram for the purpose of managing a spatial relationships between marine vessels to aid the human navigators (see Fig. 1), or the work of Ferrez – [3] which uses a kinetic regular triangulation to simulate the behavior of a granular material within a container. This particular KDS was used in order to detect the collisions between grains of various sizes (the different radii of the grains were reflected by altering the weights of the points in the triangulation).

Another field of application of KDS is the area of crowd simulation. If the crowd is simulated in the agent-based fashion, the spatial relationship among pedestrians may be managed by a kinetic data structure at the local level (i.e., to prevent collisions between pedestrians) – see [10].

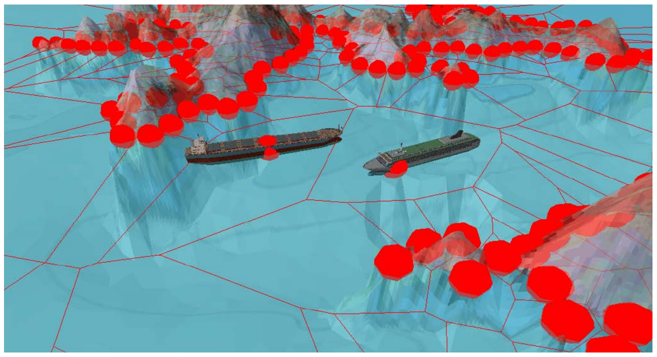


Figure 1. An example of the collision detection application of kinetic Voronoi diagram in the marine environment, [9].

The area of mathematical and physical simulations may also benefit from using KDS. Beni adressed the problem of solving partial differential equations in [6] and suggested that KDS may be used for the purposes of fluid dynamics simulations in 3D.

Although there are various types of kinetic data structures and the mathematical properties explained in this text are common to all of them, we will often explain them on the specific case of kinetic Delaunay triangulation (KDT) which is a typical and the most widely used KDS.

B. Mathematical Properties of KDS

Several different papers and theses have discussed the problem of mathematical properties of the kinetic data structures, namely the problem of computational complexity of KDS – see [2] and the problem of speedup by making the necessary computation more efficient or reducing their amount – see [5], [11]. Since the KDS management is often strongly dependent on the method of the computation of the kinetic events, which is commonly in the form of solving polynomial equations (see further), various methods for this particular task have been considered. Because these polynomials are most often nontrivial (with degree at least four) it is not practical or even not possible to solve them analyticaly. The methods most commonly used for solving these equations are the eigenvalue methods [12], methods based on interval arithmetic [5], [11] or various kinds of hybrid methods [11]. Since the computation is highly time-consuming, it is often convenient to speed up the process by exploiting various features of the polynomial functions. For this purpose, such tools as the Sturm sequences of polynomials or Descartes' rule of signs are sometimes employed, allowing us to separate the roots more effectively [13]. Another method of simplifying the computation is to replace the original polynomial equations with different equations with the same positions of the roots and their multiplicities reduced to one as suggested in [11]. In this text we will show that this modification is not correct and may lead to distortions in the topology of the managed KDS if the further root management is based on the polynomial equations with reduced root multiplicities. As far as the root location goes, this method of simplification is possible, however, we will show that finding the roots with even multiplicities may be ommited entirely as they are not necessary for the KDS management.

III. Kinetic Data Structures

In order to preserve the properties of a data structure for the mobile data, a *flight plan* is added to the primitives in the construction set – a continuous motion function which describes its movement. Furthermore, a means of computation of the events is implemented and a queue for storing these events is utilized. The trajectory of each primitive is therefore described by a function (often restricted to a function type such as real polynomial) for which we are able to compute and sort its roots (that determine the aforementioned events). The properties (as well as the use) of the queue may be found in [6]; they are not discussed here as they are not important for our purposes.

A. Predicates and Certificates

Definition 1 (Predicate): Let us have a set $\mathbf{P} = \{P_1, P_2, \ldots, P_n\}$ of n primitives, a data structure $\mathbf{DS}(\mathbf{P})$ constructed over these primitives and a finite set of discrete values $\mathbf{V} \subset \mathbb{R}$. The following function:

$$p : p(\bar{\mathbf{P}}) \rightarrow \mathbf{V} \tag{1}$$

where $\bar{\mathbf{P}} \subset \mathbf{P}$ is a subset of the set of the primitives of a given (pre-defined) size, is called a predicate of $\mathbf{DS}(\mathbf{P})$.

An example of a predicate may be for instance the orientation test or incicrcle test, see later. The predicates often consider only the location of the primitives, but it is not a general rule (e.g., the point weights are considered in the case of regular triangulation). The predicates *define* the data structures as they are used by the *certificates* to determine the correctness of these structures:

Definition 2 (Certificate): The evaluation of a predicate function p (as defined earlier) for a given subset $\bar{\mathbf{P}} \in \mathbf{P}$ in a given data structure $\mathbf{DS}(\mathbf{P})$ is called a certificate. With respect to the parameters of p and $\mathbf{DS}(\mathbf{P})$, a certificate may either yield a *failure*, a *correct state* or a *singular state*.

Let us show an example of the *incircle test* – a predicate for a Delaunay triangulation $\mathbf{DT}(\mathbf{P})$ over a given set of n points in Euclidean plane $\mathbf{P} = \{p_1, p_2, \ldots, p_n\}$ (where $p_i = [x_i, y_i]$) [14] and the associated certificate. For the sake of simplicity, let us assume that all the triangles in the triangulation are oriented counter-clockwise.

$$I(p_i, p_j, p_k, p_l) = \\ = \operatorname{sgn}\left(\det\begin{bmatrix} x_i & y_i & x_i^2 + y_i^2 & 1 \\ x_j & y_j & x_j^2 + y_j^2 & 1 \\ x_k & y_k & x_k^2 + y_k^2 & 1 \\ x_l & y_l & x_l^2 + y_l^2 & 1 \end{bmatrix}\right) \tag{2}$$

where $p_i, p_j, p_k, p_l \in \mathbf{P}$ and $\mathbb{V} = \{-1, 0, 1\}$. The function I determines the position of a point against a circumcircle of a triangle given by the other three points. We may see that it satisfies the formula given by (1). The associated certificate is then defined in such a way that it yields a *correct state* if the result of (2) is -1 (point p_l lies outside the circumcircle of the triangle given by $p_i p_j p_k$), a *singular state* if the result is 0 (point lies on the circle) and a *failure* if the result is 1 (point lies inside the circumcircle).

Definition 3 (Certificate function, certificate failure): Let us have a kinetic data structure $\mathbf{KDS}(\mathbf{P^k})$ defined on a set of n kinetic primitives $\mathbf{P^k} = \{P_1^k(t), P_2^k(t), \ldots, P_n^k(t)\}$ (i.e., their properties are functions of time $t \in \mathbb{R}$). Let us have a subset $\bar{\mathbf{P}}^\mathbf{k} \subset \mathbf{P^k}$ of a given predefined size as in Def. 1. The *certificate* function is then defined as:

$$c : c(t, \bar{\mathbf{P}}^\mathbf{k}) \to \mathbb{R} \tag{3}$$

Let us assume that c yields a correct state, a singular state and a failure $\iff$ $c > 0, c = 0, c < 0$ respectively. Given a time value $t_f \in \mathbb{R}$ and a subset $\mathbf{P_f^k} \subset \mathbf{P^k}$ for which $c(t_f, \mathbf{P_f^k}) = 0$ and there are $\varepsilon_1, \varepsilon_2 > 0$ such that $\forall t_1 \in (0, \varepsilon_1), t_2 \in (0, \varepsilon_2) : c(t_f - t_1, \mathbf{P_f^k}) > 0 \wedge c(t_f + t_2, \mathbf{P_f^k}) < 0$, we call t_f a time of certificate failure.

As we may see, unlike (1), the image of function (3) $\mathcal{I}(c) = \mathbb{R}$. This is caused by the fact that the certificate functions are used not only to determine whether the KDS is in a correct state but also to determine the time instants when a certificate failure occurs (if any). Therefore, the most important mathematical challenge in the management of a kinetic data structure is the task of computing the roots of the certificate functions.

B. Primitives Movement

As stated before, the kinetization of a data structure is based on the fact that some properties of the underlying primitives become a function of time. The most straighforward example of this behavior is of course movement – for instance a Delaunay triangulation constructed over a set of points which move over time becomes a kinetic Delaunay triangulation. An example of non-moving data may be for example sorting of time-dependent values.

It is obvious that in reality, the movement may follow virtually any type of trajectory, but in the kinetic data structures we usually limit the movement to polynomial type trajectories in order to keep the computations managable. If some type of a more complex trajectory is required by the application, it is usually approximated by piecewise-polynomial curves.

C. Kinetic Delaunay Triangulation

Let us now consider the special case of kinetic $\mathbf{DT}(\mathbf{P})$: we will use a kinetic data set $\mathbf{P^k} = \{p_1(t), p_2(t), \ldots, p_n(t)\}$ where $p_i(t) = [x_i(t), y_i(t)]$ is a point in the Euclidean plane with coordinates being functions of time. We may see that since the used data structure is a (kinetic) Delaunay triangulation, the associated certificate is the incircle test. However, since the coordinates of the generating points are functions of time, the incircle test itself becomes a function of time, thus forming a certificate function:

$$I^k(p_i(t), p_j(t), p_k(t), p_l(t)) = \\ = \det \begin{bmatrix} x_i(t) & y_i(t) & x_i^2(t) + y_i^2(t) & 1 \\ x_j(t) & y_j(t) & x_j^2(t) + y_j^2(t) & 1 \\ x_k(t) & y_k(t) & x_k^2(t) + y_k^2(t) & 1 \\ x_l(t) & y_l(t) & x_l^2(t) + y_l^2(t) & 1 \end{bmatrix} \tag{4}$$

The exact properties of (4) depend solely on the nature of the movement of the kinetic primitives. If we only allow movement along polynomial trajectories, function (4) will become a polynomial function. The roots of this function will determine the time instants when any four points from $\mathbf{P^k}$ become cocircular and the data structure reaches a state of certificate failure.

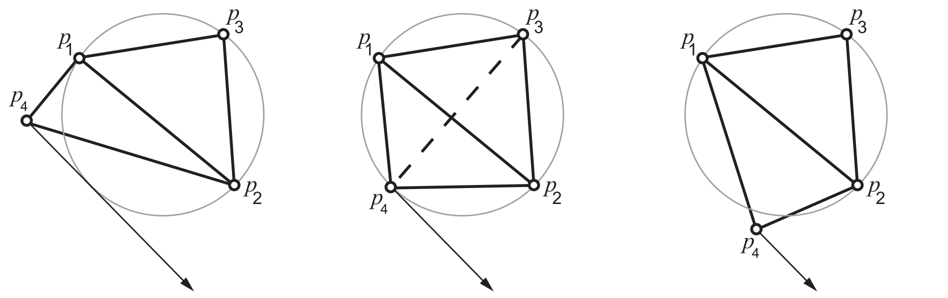
IV. Root Classification

Note that not all of the roots of a certificate function are usable as time instants of topologic event occurance. These roots are easily recognizable by considering their multiplicities. Some of the existing solutions to the KDS management problem [11] recommend that prior to the actual root computation, their method replaces each certificate function $c(t)$ with a polynomial function $d(t)$ which has the same roots but all with multiplicities equal to one. This approach is not correct, as we may see in Fig. 2.

In this figure, we may see that the point p_4 moves tangentially to the circumcircle of the triangle $p_1 p_2 p_3$ (which is not moving). There is only one (double) root of the certificate function $c(t)$ which corresponds to the time instant when the triangulation reaches the singular state depicted in Fig. 2(b) (for this single time instant, both possible triangle configurations are Delaunay-legal). This situation is the simplest possible case of this type of event and it is recognizable by obtaining a double root of the certificate function. If we reduce the root multiplicity and schedule a single event for this time, the triangulation will cease to be Delaunay and eventually it may even cease to be a triangulation. Therefore, it is necessary to be able to recognize these cases. We may see that there is no certificate failure, because there is no time period for which the associated certificate function would yield a failure. According to Def. 3 the existence of such a time period is a necessary condition for a certificate failure. The following lemma shows how these situations can be detected for polynomial certificate functions:

Lemma 1: When determining the times of topologic events, the roots of a polynomial certificate function $c(t)$ may be divided into two groups as follows:

- Roots of even multiplicity may be ignored.
- Roots of odd multiplicity determine the time of a single topologic event.



(a) Initial situation. (b) Singular state. (c) Point p_4 moves on; topology does not change.

Figure 2. Three phases of tangential movement of a point against the circumcircle of a triangle.

Proof: Let us rewrite the polynomial $c(t)$ as:

$$c(t) = (t - t_0)^r \cdot q(t) \tag{5}$$

where t_0 is a root of $c(t)$ with multiplicity r and $q(t)$ is a polynomial function. Let us now find an arbitrary small $\varepsilon > 0$ such that there will be an interval $I = (t_0 - \varepsilon; t_0 + \varepsilon)$ such that $q(t)$ has no roots in I (i.e., r is the maximum positive integer that satisfies the formula); let $c_0(t) = (t - t_0)^r$.

If t_0 is of even multiplicity, we may say that $r = 2k$ and thus:

$$c_0(t) = \left[(t - t_0)^2\right]^k \tag{6}$$

We may see that $\forall t \in \mathbb{R} : c_0(t) \geq 0$ and since $q(t)$ does not have any roots in I (and thus the sign of its value does not change over I because it is a continuous function), we may clearly see that the sign of $c(t)$ does not change over I (if the zero at $t = t_0$ is ignored). Moreover, we may see that the certificate with the certificate function $c(t)$ does not fail for any time $t \in I$ and since t_0 is the only root of $c(t)$ in I, it does not mark a certificate failure.

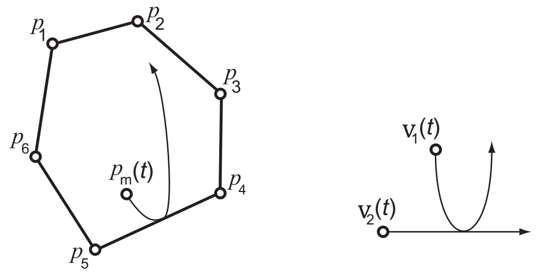
If t_0 is of odd multiplicity, then $r = 2k + 1$ and we have:

$$c_0(t) = \left[(t - t_0)^2\right]^k \cdot (t - t_0) \tag{7}$$

We may see that the sign of $c_0(t)$ does change exactly once in the interval I and thus the sign of $c(t)$ changes too and the certificate function fails, determining the time of a single topologic event. ■

Lemma 1 shows that the multiplicities of the roots of $c(t)$ need to be taken into consideration when handling the kinetic data structures. If we, for instance, replace the certificate function $c(t)$ with other polynomial function $d(t)$ which has the same roots but all with multiplicities equal to one, we will incorrectly obtain a certificate failure for the cases similar to the one shown if Fig. 2(b). This would cause a nonexistent topologic event to be executed and lead to a topology distortions upon the event execution (thus breaking the precondition of having a specific data structure). Finaly, due to these errors, the whole process of managing the KDS will probably become unstable as the distorted topology may cause scheduling even more nonexistent events until it is virtually impossible (or at least meaningless) to handle the damaged data structure. Using the information in Lemma 1, we may safely ignore this case and avoid the damage to the data structure.

The situation depicted in Fig. 2 is specifically related to the special case of the kinetic Delaunay triangulation and, as stated before, it is recognizable by obtaining a double root of the certificate function. For different kinetic data structures, similar examples may include a point moving tangentially towards the convex hull of the data set for kinetic convex hull management or comparison of such time-dependent values as $v_1(t) = t^2$ and $v_2(t) = 0$ for kinetic sorting as shown in Fig. 3. Note that these situations are also mathematically recognizable by finding double roots of the associated certificate functions.



(a) Kinetic convex hull. (b) Kinetic sorting.

Figure 3. More examples of KDS configurations with roots of even multiplicity.

V. Discussion

Lemma 1 shows that it is necessary to distinguish between the types of roots (given by their multiplicities) obtained during KDS management since not all of them are viable for topologic event determination. Even though the principle was presented on the example of kinetic Delaunay triangulation, the same situation will occur for any type of kinetic data structure if the time change of the primitives is described by polynomial equations.

However, in practical applications, the limited precision of floating-point representation of numbers will most probably effectively prevent vast majority of these cases from occuring by slightly altering the polynomials in such a way that the multiple roots will be separated. In such cases, the roots will be processed in the usual way one after another and the resulting data structure *may* stay correct. However, this feature cannot be relied on, especially for the roots of multiplicities greater than two, and it is obviously numericaly more stable to try to avoid these cases completely.

Even though the presented theory is demonstrated only on the example of kinetic Delaunay triangulation, it is applicable on any type of kinetic data structure that is based on polynomial certificate functions since Lemma 1 only considers the multiplicities of the roots of a polynomial certificate function and is thus independent on the specific meaning of these equations (and the nature of the parenting kinetic data structure).

VI. Conclusion

We show that it is not mathematically correct to simplify the polynomial certificate equations by reducing the multiplicities of all of their roots to one. It is not correct if done in order to speed up the computation by processing them as simple roots regardless of their original multiplity during the KDS management. The presented theory shows that this simplification is only valid for roots of odd multiplicity. The roots of even multiplicity should either preserve even multiplicity or they should be removed entirely because they provide us with no information about the changes in the kinetic data structure. This would also simplify the certificate equation even further.

Acknowledgement

The work was supported by the *UWB grant SGS-2010-028 Advanced Computer and Information Systems* and by *Ministry of Education project Kontakt II – LH11006 INGEM – Interactive Geometric Models for Simulation of Natural Phenomena and Crowds*.

References

[1] J. Basch, L. J. Guibas, and J. Hershberger, "Data structures for mobile data," *Journal of Algorithms*, vol. 31, no. 1, pp. 1–28, 1999.

[2] G. Albers, L. J. Guibas, J. S. B. Mitchell, and T. Roos, "Voronoi diagrams of moving points," *International Journal of Computational Geometry and Applications*, vol. 8, no. 3, pp. 365–380, 1998. [Online]. Available: citeseer.ist.psu.edu/albers95voronoi.html

[3] J.-A. Ferrez, "Dynamic triangulations for efficient 3d simulation of granular materials," Ph.D. dissertation, cole Polytechnique Fdrale De Lausanne, 2001.

[4] M. Gavrilova, J. Rokne, and D. Gavrilov, "Dynamic collision detection in computational geometry," in *12th European Workshop on Computational Geometry*, Munster, Germany, 1996, pp. 103–106.

[5] D. Russel, "Kinetic data structures in practice," Ph.D. dissertation, Stanford, CA, USA, 2007, adviser-Guibas, Leonidas.

[6] L. H. Beni, "Development of a 3d kinetic data structure adapted for a 3d spatial dynamic field simulation," Ph.D. dissertation, Université Laval, Québec, 2009.

[7] J. Erickson, L. J. Guibas, J. Stolfi, and L. Zhang, "Separation-sensitive collision detection for convex objects," in *SODA '99: Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1999, pp. 327–336.

[8] L. J. Guibas, F. Xie, and L. Zhang, "Kinetic collision detection: Algorithms and experiments," in *ICRA*, 2001, pp. 2903–2910.

[9] I. R. Goralski and C. M. Gold, "Maintaining the spatial relationships of marine vessels using the kinetic Voronoi diagram," in *ISVD '07: Proceedings of the 4th International Symposium on Voronoi Diagrams in Science and Engineering*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 84–90.

[10] S. Goldenstein, M. I. Karavelas, D. N. Metaxas, L. J. Guibas, E. Aaron, and A. Goswami, "Scalable nonlinear dynamical systems for agent steering and crowd simulation," *Computers & Graphics*, vol. 25, no. 6, pp. 983–998, 2001.

[11] L. J. Guibas and M. I. Karavelas, "Interval methods for kinetic simulations," in *SCG '99: Proceedings of the fifteenth annual symposium on Computational geometry*. New York, NY, USA: ACM, 1999, pp. 255–264.

[12] K. W. Ellenberger, "Algorithm 30: numerical solution of the polynomial equation," *Commun. ACM*, vol. 3, no. 12, p. 643, 1960.

[13] A. Ralston, *A First Course in Numerical Analysis*. McGraw-Hill, Inc.: New York, 1965.

[14] O. Hjelle and M. Dæhlen, *Triangulations and Applications*. Berlin Heidelberg: Springer, 2006.