Journal of Computational and Applied Mathematics 251 (2013) 8–21

Contents lists available at SciVerse ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Algorithms for projecting points onto conics

N. Chernov [a,*], S. Wijewickrema [b]

[a] Department of Mathematics, University of Alabama at Birmingham, Birmingham, AL 35294, USA

[b] Department of Otolaryngology, University of Melbourne, 32 Gisborne St, East Melbourne VIC 3002, Australia

ARTICLE INFO

Article history:
Received 24 September 2012
Received in revised form 12 March 2013

MSC:
41A10
65S05
65D10

Keywords:
Least squares fitting
Orthogonal projection
Ellipses
Conics
Cubic equations
Quartic equations

ABSTRACT

We study the problem of projecting 2D points onto quadratic curves (ellipses, hyperbolas, parabolas). We investigate various projection algorithms focusing on those that are mathematically proven to produce (or converge to) correct results in all cases. Our tests demonstrate that those may be still unfit for practical use due to large computational errors. We present two new algorithms that are not only theoretically proven to converge, but achieve nearly perfect accuracy.

1. Introduction

A popular task in computer vision and statistics is fitting curves to data points. Generally, one can fit curves in many ways—by Hough transform [1], by constructing principal curves [2], by minimax or minisum criteria [3], or by the classical least squares method.

Most of these methods minimize, in various senses, geometric distances from the fitting curve to the given points $(x_1, y_1), \dots, (x_n, y_n)$. For example, the least squares fit minimizes

$$\sum_{i=1}^{n} d_i^2 = \sum_{i=1}^{n} (x_i - x_i')^2 + (y_i - y_i')^2 \to \min. \tag{1}$$

Here d_i denotes the distance from (x_i, y_i) to the fitting curve and (x_i', y_i') the (orthogonal) projection of (x_i, y_i) on the curve. Thus finding the projections of the given points on a curve is an integral part of problem (1).

First publications on the least squares fit (1) date back to the 1870s [4,5]. In the simplest case – fitting straight lines to points – the problem has an analytic solution, though it is not completely trivial; see some history in [6, Chapter 1]. Fitting circles to points was first mentioned in the 1950s [7,8]. This problem does not have a closed-form solution and can only be solved by iterative approximations [6, Chapter 4]. However projecting given points onto a circle is a simple task that has an elementary solution.

* Corresponding author. Tel.: +1 205 934 2154; fax: +1 205 934 9025.
E-mail addresses: chernov@math.uab.edu, chernov@uab.edu (N. Chernov), sudanthi.wijewickrema@unimelb.edu.au (S. Wijewickrema).

0377-0427/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.cam.2013.03.031

Since the 1970s, researchers in various areas have been fitting ellipses and hyperbolas (i.e., conic sections, or *conics*) to points [9–12]. This task has recently gained major importance in the computer vision community, for image processing and pattern recognition applications.

The conic fitting problem does not have a closed-form solution. Moreover, there is no simple algorithm for projecting given points onto a conic. For these reasons, until the mid-1990s the minimization of geometric distances (1) was not used for the purpose of fitting conics. Instead, researchers replaced geometric distances d_i with various (easily computable) algebraic distances and solved the resulting minimization problem [11,13–15]. Such approximations were fast but often unsatisfactory [16,11,17,18]. Recent research [19] shows that algebraic distance minimization methods have consistently larger statistical errors than those of geometric minimization methods (1).

The first fundamental work on fitting ellipses to data by minimizing the geometric distances (1) was published in 1994 by Gander et al. [17]. They did not find an efficient projection algorithm; they minimized (1) by various roundabout ways avoiding explicit calculation of the projected points (x_i', y_i'). All their algorithms were complicated, slow and often diverged. The paper [17] made a long lasting impression that fitting ellipses by the minimization of (1) was a prohibitively difficult, almost hopeless task.

Alternative approaches were found by Ahn et al. in 2001 [20–22,16]. They used iterative projection algorithms (to be described below in Section 3) and developed efficient fitting procedures. Ahn classified fitting schemes into several categories [16], of which *total* methods avoided the computation of the projected points, but for this reason tended to be overly complicated, slow, and prone to divergence. On the contrary, *variable-separation* methods involve the computation of the projected points and are fast and reliable. Those are further divided into *coordinate-based* fits and *distance-based* fits, of which the latter are slightly more efficient (see also [23]).

Ahn's best general fitting scheme (variable-separation and distance-based) can be applied to arbitrary curves in 2D and surfaces in 3D; the latter may be described by explicit equations, implicit equations, or parametric equations. More work in this direction was done recently by Aigner et al. [24], Sturm et al. [25], Wijewickrema et al. [26], and Chernov et al. [23,27].

The efficiency of the variable-separation and distance-based fitting scheme is now widely recognized and it is well optimized and streamlined. Its "weakest link" remains the computation of the projected points. Ahn repeatedly pointed out that this step is "the time-consuming part of the variable-separation method"; see e.g., [22] and [16, p. 56]. In our experience, it indeed often demands more computational time than the rest of the fitting procedure.

Besides, the efficiency and reliability of the projection methods have not been studied systematically. Some researchers propose heuristic iterative schemes [20–22,16] that are relatively fast, but their convergence is not guaranteed (and occasionally they do diverge). On the other hand, certain theoretically reliable methods exist [28,29,26] but their practical performance remains virtually unexplored. This is what we investigate in this paper.

We restrict our analysis to theoretically proven methods, i.e., those which are mathematically proven to produce (or converge to) the correct result in *all* cases. Of course, theoretical proof assumes that there are no round-off errors. In machine computations, even theoretically exact methods may return unsatisfactory (or catastrophic) results due to numerical errors. Thus, we also ensure that these methods are implemented in such a way so as to minimize numerical errors.

We will demonstrate that some theoretically precise methods tend to be unreliable in practice and fail serious numerical tests. But more importantly, we present two algorithms that can be implemented with proper care to produce very accurate results (to machine precision) in virtually all cases. At the same time we ensure that their speed is comparable to that of fast unreliable heuristic schemes.

In the modern computational world, reliability and accuracy take priority over sheer speed. For example, solving least squares problems by the fast and numerically unstable method of normal equations is almost universally abandoned—slower but more accurate and reliable QR and SVD solutions are used instead [30]. The logic is simple—with the ever increasing speed of modern CPUs and the availability of highly optimized library functions for matrix operations it makes little sense to save milliseconds of computational time while sacrificing accuracy and jeopardizing reliability. We adopt these same standards here in our study of projection algorithms.

The paper is organized as follows. We will first review known projection algorithms, old and very recent. Then we identify those capable of achieving higher accuracy and reliability and discuss their implementations. Lastly we compare all these methods in a series of numerical tests.

2. Quartic equation method

Our goal is to project a point (u, v) onto a conic

$$Q(x, y) := Ax^2 + 2Bxy + Cy^2 + 2Dx + 2Ey + F = 0. \tag{2}$$

The projection must be orthogonal to the conic, i.e., the vector $(u-x, v-y)$ must be parallel to the gradient vector $\nabla Q(x, y)$. Thus the projection point (x, y) satisfies (2) and the orthogonality conditions

$$u - x = t(Ax + By + D) \quad \text{and} \quad v - y = t(Bx + Cy + E) \tag{3}$$

for some real t.

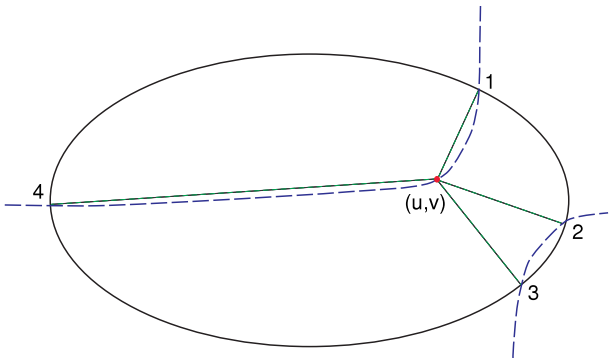

Fig. 1. Four orthogonal projections of a point (u, v) on an ellipse. The footpoint 1 is the closest projection. The auxiliary conic here is a hyperbola (shown by blue dashed lines). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Solving the linear system (3) for x and y gives

$$x = \frac{(u - Dt)(Ct + 1) - (v - Et)Bt}{(At + 1)(Ct + 1) - (Bt)^2} \tag{4}$$

and

$$y = \frac{(v - Et)(At + 1) - (u - Dt)Bt}{(At + 1)(Ct + 1) - (Bt)^2}. \tag{5}$$

Substitution of these expressions into (2) gives a polynomial equation of degree four (*quartic equation*) in t:

$$c_4t^4 + c_3t^3 + c_2t^2 + c_1t + c_0 = 0, \tag{6}$$

where $c_0, \dots, c_4$ can be readily expressed in terms of A, B, C, D, E, F and u, v. Solving Eq. (6) for t gives us up to four real roots. Substituting their values back into (4) and (5) yields up to four points (x_j, y_j), $1 \le j \le J$, where $J \le 4$ denotes the number of points.

These points (x_j, y_j) are the footpoints of the orthogonal projections of the original point (u, v) on the conic (2). Fig. 1shows that there may be indeed up to four such orthogonal projections (though usually only one of them is the nearest point we are looking for). The nearest point corresponds to the smallest value of $(u - x_j)^2 + (v - y_j)^2$.

Now how do we solve (6)? There exist analytic formulas for the roots of a quartic, thus our projection problem has an analytic solution which gives the exact result. It is however unsatisfactory for a number of reasons.

First, the analytic solution is long and complicated. It involves complex numbers at various intermediate steps that one has to go through even if all the roots of (6) are real. Next, a simple flop count shows that the above analytic solution is not even that fast. It is slower than, say, Eberly's method (Section 6), which is also mathematically proven to give correct results.

Most importantly, the analytically computed solution of (6) is known to be numerically unstable. Precisely, there are five classical solutions of quartic equations: due to Descartes–Euler–Cardano [31], Christianson–Brown [32], Ferrari–Lagrange [33], Neumark [34], and Yacoub–Fraidenraich–Brown [35]. The stability (or instability) of these solutions depends on the signs of the coefficients $c_0, \dots, c_4$ and certain auxiliary variables. Some methods are stable for *some* combinations of signs, but none is stable for *all* combinations. Moreover, for the majority of combinations of signs (by some counts, 12 out of 16; see [36]) *none* of these methods is stable; see a detailed analysis in [36] and a recent work [37].

The numerical instability of the analytic solution of quartics has a bad reputation in the scientific community [20]. For this reason the above method is virtually never used in practice (in fitting applications). We still include it in our numerical tests in Section 8 to show how unreliable it is.

3. Newton's iterations

This is a heuristic approach but it has its merits.

Upon elimination of t from (3) we get

$$R(x, y) := (u - x)(Bx + Cy + E) - (v - y)(Ax + By + D) = 0. \tag{7}$$

Now we need to solve the system of two Eqs. (2) and (7) for two unknowns, x and y. We can apply a standard Newton iterative process

$$x_{k+1} = x_k + dx_k \quad \text{and} \quad y_{k+1} = y_k + dy_k \tag{8}$$

where

$$\begin{bmatrix} dx_k \\ dy_k \end{bmatrix} = - \begin{bmatrix} Q_x & Q_y \\ R_x & R_y \end{bmatrix}^{-1} \begin{bmatrix} Q \\ R \end{bmatrix}. \tag{9}$$

Here Q_x, Q_y, R_x, R_y denote the partial derivatives of Q and R with respect to the respective variables, and all the values of Q, R, and their derivatives are evaluated at the point (x_k, y_k).

The rate of convergence of Newton's iterations is known to be fast (quadratic), but the process may converge to a wrong limit point or diverge if the starting point (x_0, y_0) is chosen poorly. A natural choice for the initial guess is the original point, i.e., $(x_0, y_0) = (u, v)$; see [16]. If the point (u, v) is close enough to the conic, the iterations converge fast and yield the correct result. But if (u, v) is far from the conic, the iterations progress slowly, may converge to the wrong limit, or diverge altogether.

In our tests, where conics and points were generated randomly (Section 8), this procedure failed in 3% of the cases. While 97% success rate is very high, the method cannot be regarded as totally reliable. Ahn [16] designed certain modifications to speed up the convergence and reduce the failure rate, but he admitted that success could not be guaranteed.

While it is not our purpose to investigate heuristic methods here, this one has its merits. It can be used to "polish" solutions found by other (reliable) algorithms. Once a projection point (x', y') is found somehow, one can use it to start the Newton process (8)–(9), i.e., set $(x_0, y_0) = (x', y')$, and make a few iterations. This helps reduce numerical errors in difficult cases.

But even as a "polishing tool" the above method should not be used without an additional safety control. In our tests we have computed

$$E_k := |Q(x_k, y_k)| + |R(x_k, y_k)| \tag{10}$$

and stopped iterations once E_k started growing, i.e., once $E_{k+1} > E_k$.

4. Wijewickrema–Esson–Papliński method

A new projection method was proposed recently by Wijewickrema, Esson, and Papliński; we will abbreviate it as WEP. It was presented at a conference [26] but never published in a journal. Our presentation here differs from the original one [26] in some details; we focus more on practical issues.

The quadratic equation (2) can be written in matrix form as

$$\mathbf{z}^T \mathbf{M} \mathbf{z} = \mathbf{0}, \tag{11}$$

where $\mathbf{z} = [x\ y\ 1]^T$ and

$$\mathbf{M} = \begin{bmatrix} A & B & D \\ B & C & E \\ D & E & F \end{bmatrix}. \tag{12}$$

Eq. (7) is also quadratic in x and y and can be written as

$$\mathbf{z}^T \mathbf{N} \mathbf{z} = \mathbf{0}, \tag{13}$$

where

$$\mathbf{N} = \begin{bmatrix} -2B & A-C & uB-vA-E \\ A-C & 2B & uC-vB+D \\ uB-vA-E & uC-vB+D & 2(uE-vD) \end{bmatrix}. \tag{14}$$

Eq. (13) defines another conic in the xy plane (called *auxiliary conic*). The top left 2×2 block $\mathbf{N}_{2,2}$ of $\mathbf{N}$ has a non-positive determinant

$$\det \mathbf{N}_{2,2} = \det \begin{bmatrix} -2B & A-C \\ A-C & 2B \end{bmatrix} = -4B^2 - (A-C)^2 \le 0.$$

Thus the auxiliary conic (13) can be of the following types:

1. a hyperbola (whenever $\det \mathbf{N} \neq 0$ and $\det \mathbf{N}_{2,2} < 0$);
2. a pair of intersecting lines (whenever $\det \mathbf{N} = 0$ and $\det \mathbf{N}_{2,2} < 0$);
3. a single line (whenever $\mathbf{N} \neq \mathbf{0}$ and $\det \mathbf{N}_{2,2} = 0$);
4. undefined ($\mathbf{N} = \mathbf{0}$, i.e., $\mathbf{N}$ is a zero matrix).

Case 3 occurs if and only if the original conic (11) is a circle and the given point (u, v) is *not* its center. Case 4 occurs if and only if the original conic (11) is a circle and (u, v) *is* its center. We will assume that the original conic (11) is not a circle, as in that case the projection point can be easily found by elementary geometry.

Now all the footpoints of the orthogonal projections of the given point (u, v) on the given conic (11) (recall that there is at least one and at most four of them; cf. Section 2) satisfy (13) as well. Thus those points are exactly the points of intersection of our two conics (11) and (13); see Fig. 1.

In order to find the points of intersection of the conics (11) and (13) we consider a family (a *pencil*) of conics

$$\mathbf{z}^T (\alpha \mathbf{M} + \beta \mathbf{N}) \mathbf{z} = \mathbf{0}, \tag{15}$$

where α and β are real parameters. Any conic in this pencil obviously passes through the intersection points of (11) and (13).

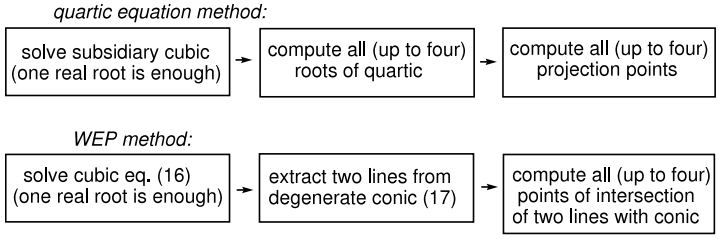

Fig. 2. Main steps of the quartic equation method and the WEP.

The above pencil contains at least one (and at most three) degenerate conics; see [38]. Our next goal is to find one of them. If the original conic (11) is degenerate, i.e., if $\det \mathbf{M} = 0$, then we just set $\beta = 0$. In that case our conic (11) consists of a pair of lines (or a single line), and the projection problem can be easily solved by extracting the individual lines from the conic and reducing the problem to the projection of a point onto lines. The process of obtaining individual lines from a degenerate conic is discussed later in this section and in more detail in Section 5.

If $\det \mathbf{M} \neq 0$, we set $\beta = 1$ and solve the equation

$$\det(\alpha \mathbf{M} + \mathbf{N}) = 0 \tag{16}$$

for α. Its leading term is $(\det \mathbf{M})\alpha^3$, and since $\det \mathbf{M} \neq 0$, it is a cubic equation. So it has at least one and at most three real roots. Let α be one of them. If there are three real roots, we recommend that the one most distant from the other two is chosen, to avoid ill-conditioned roots. However, if that root is too close to zero, it may be ill-conditioned, too. In that case one usually makes the change of variable $\zeta = 1/\alpha$ and solves the resulting cubic equation for ζ, finds its largest root and then computes $\alpha = 1/\zeta$; see [39]. In our case this is equivalent to solving equation $\det(\zeta \mathbf{N} + \mathbf{M}) = 0$, which effectively means interchanging the matrices $\mathbf{M}$ and $\mathbf{N}$ in (16). This is done, in a more consistent way, in the next section.

Now denote $\mathbf{E} = \alpha \mathbf{M} + \mathbf{N}$ with α satisfying (16). One can prove that $\mathbf{E} \neq \mathbf{0}$, i.e., $\mathbf{E}$ cannot be a zero matrix. Since every conic in the pencil (15) passes through the points of intersection of the conics (11) and (13), the conic defined by the matrix $\mathbf{E}$, i.e.

$$\mathbf{z}^T \mathbf{E} \mathbf{z} = 0, \tag{17}$$

passes through those points as well.

Since $\det \mathbf{E} = 0$, conic (17) is degenerate. In addition, one can prove that $\mathbf{E}$ cannot have two positive or two negative eigenvalues, thus conic (17) cannot be a single point. Therefore conic (17) consists of two lines or a single line. Let $a_i x + b_i y + c_i = 0$ be the equations of those two lines, where $i = 1, 2$. Denote $\mathbf{u}_i = [a_i\ b_i\ c_i]^T$. If these two lines coincide, we have $\mathbf{u}_1 = \mathbf{u}_2$. Then one can prove that

$$\mathbf{E} = \mathbf{u}_1 \mathbf{u}_2^T + \mathbf{u}_2 \mathbf{u}_1^T \tag{18}$$

up to a scalar multiple. In fact, every 3×3 symmetric singular matrix $\mathbf{E}$ whose two nonzero eigenvalues have opposite signs can be decomposed by (18) with some vectors $\mathbf{u}_1, \mathbf{u}_2$.

We discuss practical methods for finding $\mathbf{u}_1$ and $\mathbf{u}_2$ in the next section.

It remains to find the points of intersection of the original conic (2) with the lines $a_i x + b_i y + c_i = 0$. A simultaneous solution of the quadratic equation (2) with the linear equation $a_i x + b_i y + c_i = 0$ reduces to a quadratic equation in one variable; it gives us up to two real roots. Thus we have up to four solutions in total.

These solutions are all the footpoints (x_j, y_j), $1 \le j \le J \le 4$, of the orthogonal projections of the given point (u, v) on the conic (2); cf. Section 2. Lastly we select the footpoint with the smallest $(u - x_j)^2 + (v - y_j)^2$. This completes the WEP projection algorithm.

5. Analysis of the WEP

One cannot help noticing similarities between the WEP and the quartic equation method of Section 2. Indeed, every analytic solution of a quartic begins with forming and solving the so called *subsidiary* cubic equation [36]. Thus we can represent the two methods schematically as in Fig. 2.

As the first step of both methods, one solves the respective cubic analytically. The respective formulas can be arranged to reduce the loss of accuracy [40, Section 5.6], which is sufficient for our purposes (see Section 8). The analytic solution of cubics does not have as bad a reputation as that of quartics; in fact, some researchers seem to be more concerned with the threat of overflow than with that of catastrophic cancellations [36].

Alternatively, one can employ matrix library functions (if available) to solve the cubic equation by finding the eigenvalues of the companion matrix (this way of solving polynomial equations is standard in MATLAB).

In the WEP algorithm, one can avoid forming and solving the cubic equation (16) and instead solve generalized eigenvalue problem

$$\mathbf{N}\mathbf{v} = \lambda \mathbf{M}\mathbf{v}. \tag{19}$$

In solving (19), we only need one real eigenvalue λ, and we do not need any eigenvectors. Having λ we can compute $\mathbf{E} = \mathbf{N} - \lambda\mathbf{M}$ and continue the WEP procedure as described above.

The results can be sometimes improved if we replace (19) with

$$\mathbf{M}\mathbf{v} = \lambda'\mathbf{N}\mathbf{v}, \tag{20}$$

and then use the degenerate matrix $\mathbf{E}' = \mathbf{M} - \lambda'\mathbf{N}$ instead of $\mathbf{E} = \mathbf{N} - \lambda\mathbf{M}$. We found that one should use (19) if $\mathbf{N}$ is "more singular" than $\mathbf{M}$ and (20) otherwise. One can pick the matrix with the higher condition number as the "more singular" matrix. If the computation of condition numbers is too expensive, the determinant can be used instead—a smaller determinant indicates a "more singular" matrix.

Yet another way of solving (16) is to solve eigenvalue problems

$$\mathbf{M}^{-1}\mathbf{N}\mathbf{v} = \lambda\mathbf{v} \quad \text{or} \quad \mathbf{N}^{-1}\mathbf{M}\mathbf{v} = \lambda\mathbf{v}, \tag{21}$$

instead of (19) or (20), respectively. This option may be suitable if the available matrix library has no routine for solving the generalized eigenvalue problem.

To summarize, the first step of the WEP algorithm can be handled in four ways:

1. Solving cubic equation (16) analytically (see [40, Section 5.6]);
2. Computing the eigenvalues of the companion matrix;
3. Solving the generalized eigenvalue problem (19) or (20);
4. Solving the ordinary eigenvalue problem (21).

In our numerical experiments (Section 8) we tested all these four options and found that they achieve the same accuracy. In terms of speed, they are all comparable in MATLAB, but in C++ the first option is 5–10 times faster than the others (and it does not depend on matrix library functions).

We now turn to the second step of the WEP algorithm—computing the vectors $\mathbf{u}_1$ and $\mathbf{u}_2$ from $\mathbf{E}$. The diagram in Fig. 2 suggests that it plays a role similar to the conversion of a root of the subsidiary cubic to the roots of the quartic (6). This step is crucial, as this is where a major loss of accuracy occurs in the analytic solution of the quartic equation; see analysis in [36].

Fortunately, the WEP algorithm bypasses the above pitfalls by not forming or solving any quartic equations. It turns out that one can compute the vectors $\mathbf{u}_1$ and $\mathbf{u}_2$ from $\mathbf{E}$ safely, without significant loss of accuracy. We describe two ways for doing this below.

The first one is based on the eigendecomposition of the matrix $\mathbf{E}$. Since the latter is symmetric, we get three real eigenvalues $\lambda_1 \ge \lambda_2 \ge \lambda_3$ and the corresponding unit eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$.

Now recall that one eigenvalue is zero and the other two cannot be both positive or both negative. Thus, without loss of generality we will assume that $\lambda_2 = 0$. Then we can compute $\mathbf{u}_1$ and $\mathbf{u}_2$ as follows:

$$\mathbf{u}_1 = \lambda_1^{1/2}\mathbf{v}_1 + |\lambda_3|^{1/2}\mathbf{v}_3 \tag{22}$$

and

$$\mathbf{u}_2 = \lambda_1^{1/2}\mathbf{v}_1 - |\lambda_3|^{1/2}\mathbf{v}_3. \tag{23}$$

The second way of computing the vectors $\mathbf{u}_i = [a_i\ b_i\ c_i]^T$ $(i = 1, 2)$ is more elementary and need no matrix library functions. It uses the "deflation" of the problem. Denote $\bar{\mathbf{u}}_i = [a_i\ b_i]^T$ for $i = 1, 2$ and let $\bar{\mathbf{E}} = \bar{\mathbf{E}}_{1,2}$ denote the top left 2×2 block of the matrix $\mathbf{E}$. Then (18) implies

$$\bar{\mathbf{E}} = \bar{\mathbf{u}}_1\bar{\mathbf{u}}_2^T + \bar{\mathbf{u}}_2\bar{\mathbf{u}}_1^T, \tag{24}$$

so we get the same problem as (18) but in two dimensions. It can be solved as described above by the eigendecomposition of $\bar{\mathbf{E}}$. More precisely, if $\bar{\lambda}_1 \ge \bar{\lambda}_2$ are the eigenvalues and $\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2$ the corresponding unit eigenvectors of $\bar{\mathbf{E}}$, then

$$\bar{\mathbf{u}}_1 = \bar{\lambda}_1^{1/2}\bar{\mathbf{v}}_1 + |\bar{\lambda}_2|^{1/2}\bar{\mathbf{v}}_2$$

and

$$\bar{\mathbf{u}}_2 = \bar{\lambda}_1^{1/2}\bar{\mathbf{v}}_1 - |\bar{\lambda}_2|^{1/2}\bar{\mathbf{v}}_2.$$

Lastly c_1 and c_2 can be computed by solving two linear equations

$$a_1c_2 + a_2c_1 = E_{13} \quad \text{and} \quad b_1c_2 + b_2c_1 = E_{23}.$$

The above deflation trick can be reorganized by redefining $\bar{\mathbf{u}}_i = [b_i\ c_i]^T$ for $i = 1, 2$ and using the *bottom right* 2×2 principal minor $\bar{\mathbf{E}} = \bar{\mathbf{E}}_{2,3}$ of $\mathbf{E}$. We can also redefine $\bar{\mathbf{u}}_i = [a_i\ c_i]^T$ and use the minor $\bar{\mathbf{E}} = \bar{\mathbf{E}}_{1,3}$ formed by the 1st and 3rd rows and columns of $\mathbf{E}$.

Thus the deflation can be organized in three different ways, based on any of the three minors ($\bar{\mathbf{E}}_{1,2}$, $\bar{\mathbf{E}}_{1,3}$, or $\bar{\mathbf{E}}_{2,3}$) of the matrix $\mathbf{E}$. We found that numerical errors are minimized if one uses the "least singular" minor, i.e., the one with the largest determinant. With this choice, the accuracy of the deflation-based computation of $\mathbf{u}_1$ and $\mathbf{u}_2$ happens to be even slightly higher than that of the previous solution (22)–(23).

To summarize, each major step of the WEP method (the first and second blocks in Fig. 2) can be implemented with or without matrix library functions. With the latter, the coding is easier but the execution time in C++ may be 5–10 times longer (though in MATLAB the difference is less pronounced). The accuracy is nearly the same whether we use matrix library functions or not. Our MATLAB and C++ code is posted on the web [41].

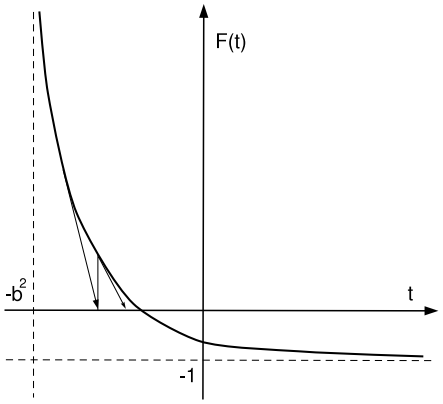

Fig. 3. A typical graph of $F(t)$ for $t > -b^2$ and the progress of Newton's iterations toward the root.

6. Eberly's projection method

The previous sections may give an idea that theoretically proven projection methods have to be overly complicated. This is not so.

A remarkably simple approach to projecting points onto ellipses was found by D. Eberly in 2004 [28, Section 14.13.1]. Not only is it simple and fast, but it comes with mathematical proof of convergence to the correct projection point in all cases, i.e., it is completely reliable. We only sketch Eberly's method here and refer to [28,23] for more details.

Suppose the given conic (2) is an ellipse. By translating and rotating the coordinate system we can represent it in canonical coordinates as

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} - 1 = 0 \quad (a \geq b > 0). \tag{25}$$

Let (u, v) be the given point in these canonical coordinates. It will be enough to project it onto the ellipse (25) and then translate and rotate the projected point back to the original coordinates.

Due to the obvious symmetry, it is enough to work in the first quadrant $u > 0$, $v > 0$; then the projection point (x, y) will also be in the first quadrant, i.e., $x > 0$, $y > 0$. Also, we exclude the degenerate cases where $u = 0$ or $v = 0$; they are fairly simple and can be handled separately (see details in [28]).

Now the orthogonality conditions (3) read

$$u - x = tx/a^2 \quad \text{and} \quad v - y = ty/b^2 \tag{26}$$

for some real t. From (26) we find

$$x = \frac{a^2 u}{t + a^2} \quad \text{and} \quad y = \frac{b^2 v}{t + b^2}. \tag{27}$$

Since $x, y > 0$, we have constraint $t > -b^2$ (recall that $a \geq b$). Substituting (27) into (25) gives a function

$$\mathcal{P}(t) = \frac{a^2 u^2}{(t + a^2)^2} + \frac{b^2 v^2}{(t + b^2)^2} - 1, \tag{28}$$

whose root we need to find (because (x, y) must lie on the ellipse). Once we solve equation $\mathcal{P}(t) = 0$ we can compute the projection point (x, y) by (27). Note that we only need to deal with *one* projection point here, unlike the previous methods that waste time on computing *all* the projection points.

Next we show how to find the root of (28). Observe that

$$\lim_{t \to -b^2+} \mathcal{P}(t) = +\infty \quad \text{and} \quad \lim_{t \to \infty} \mathcal{P}(t) = -1$$

and by direct differentiation of $\mathcal{P}$ one can see that

$$\mathcal{P}'(t) < 0 \quad \text{and} \quad \mathcal{P}''(t) > 0 \tag{29}$$

for all $t > -b^2$. Hence the function $\mathcal{P}$ is monotonically decreasing and concave; see Fig. 3. Thus standard Newton's method starting at any point t_0 where $\mathcal{P}(t_0) > 0$ will converge to the unique root of $\mathcal{P}$. A starting point can be chosen as $t_0 = \max\{au - a^2, bv - b^2\}$; see [23]. Newton's method is known to converge quadratically. In double precision it achieves the maximum possible accuracy in 5–6 iterations.

Eberly's method was extended to hyperbolas and parabolas in [23]. It was also generalized to 3D quadratic surfaces [23]. In all these cases the method seems to be the simplest and fastest among all the methods theoretically proven to produce (or converge to) the correct result.

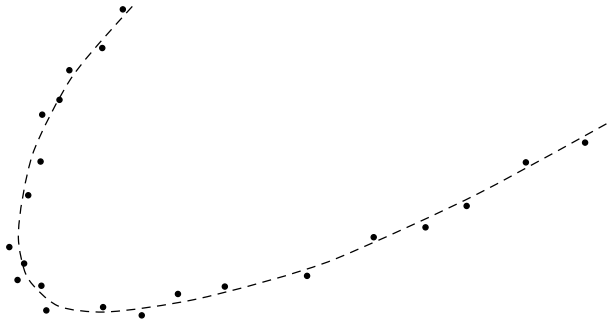

Fig. 4. A sequence of points along a parabolic arc. The best fitting ellipse (or hyperbola) may have very large axes and a very distant center.

It is not hard to "squeeze the most" out of Eberly's method, i.e., make it ultimately accurate and 100% reliable in the canonical coordinates, i.e., where the ellipse is given by (25). However the transition between the original and canonical coordinates may ruin the results.

Indeed, let (x_c, y_c) denote the center of the original ellipse. Then the coordinates of the given point (u, v) must be first translated by $u \mapsto u - x_c$ and $v \mapsto v - y_c$. Likewise, the found projection point (x, y) has to be shifted back by $x \mapsto x + x_c$ and $y \mapsto y + y_c$. If the center (x_c, y_c) is far away, i.e., its coordinates are large, then these translations will cause severe loss of precision. If $x_c, y_c \sim 10^{15}$ or larger, the results will be meaningless.

Such situations are not uncommon in fitting applications. The best fitting ellipse is usually found by iterative approximations, and those often return progressively larger ellipses that converge to a parabola. This phenomenon was noticed as early as 1979 by Bookstein [11] who wrote: "The fitting of a parabola is a limiting case, exactly transitional between ellipse and hyperbola. As the center of ellipse moves off toward infinity while its major axis and the curvature of one end are held constant…". For reasons causing such phenomena, see [42,27].

Fig. 4 shows a typical example where a set of points is best approximated by an elliptic arc which is very close to a parabola (or even to a branch of a hyperbola). Here the best fitting ellipse may have arbitrarily large axes and an arbitrarily distant center. Furthermore, small perturbations of the coordinates of the points may cause arbitrarily large changes in the ellipse's center and axes or even alter the conic type (from ellipse to parabola or hyperbola). In this case Eberly's projection method is likely to end disastrously.

On the other hand, nothing in the above example calls for a disaster. The approximating conic arc (more precisely, its part that we see in Fig. 4) is quite stable under small perturbations of the points, and the projections of the latter on the arc are stable, too. In other words, the task of projecting the points onto the conic is well-conditioned in this example. The failure of Eberly's method is entirely due to its intrinsic limitations.

Actually, if we define the conic by quadratic equation (2), rather than by its geometric parameters (center, axes, etc.), then the coefficients A, B, C, D, E, F could be reconstructed quite accurately, and they would remain stable under small perturbations of the points. This gives us a hint that Eberly's method can be improved if we work with a conic defined by algebraic equation (2) instead of its canonical coordinates.

7. Our modification of Eberly's method

As before, let the conic be given by the quadratic equation

$$Ax^2 + 2Bxy + Cy^2 + 2Dx + 2Ey + F = 0. \tag{30}$$

We assume that the vector of coefficients is normalized ($A^2 + B^2 + C^2 + D^2 + E^2 + F^2 = 1$), so that their values are of order 1. We will also assume that the coordinates u and v of the given point (u, v) are of order 1, as otherwise they can (and should) be rescaled to avoid numerical complications.

Now we can rotate the coordinate system to make $B = 0$. This is achieved by the eigendecomposition

$$\begin{bmatrix} A & B \\ B & C \end{bmatrix} = \mathbf{Q}\mathbf{D}\mathbf{Q}^T, \tag{31}$$

where $\mathbf{Q}$ is an orthogonal matrix and $\mathbf{D}$ is a diagonal matrix. The eigendecomposition of a 2×2 symmetric matrix is an elementary problem that can be solved directly without calling matrix library functions (in fact, our direct solution of (31) turns out more accurate than solving (31) by standard library functions in either C++ or MATLAB).

Then we change variables by $[xy] \mapsto [xy]\mathbf{Q}$ and respectively the coordinates of the given point by $[uv] \mapsto [uv]\mathbf{Q}$. In the new coordinates the conic's parameters change as

$$\begin{bmatrix} A & B \\ B & C \end{bmatrix} \mapsto \mathbf{D}, \qquad [DE] \mapsto [DE]\mathbf{Q}, \qquad F \mapsto F,$$

so the conic's equation in the new coordinates is

$$Ax^2 + Cy^2 + 2Dx + 2Ey + F = 0, \tag{32}$$

(note that $B = 0$). We will use the same symbols A, C, D, E, F for the coefficients of the new equation (32); this should cause no confusion.

Next by shifting $x \mapsto x - u$ and $y \mapsto y - v$ (by which we mean $x_{\text{new}} = x_{\text{old}} - u$ and $y_{\text{new}} = y_{\text{old}} - v$) we can move the given point (u, v) to the origin. This results in the following changes of the parameters:

$$D \mapsto D + Au, \qquad E \mapsto E + Cv, \qquad F \mapsto F + Au^2 + 2Du + Cv^2 + 2Ev.$$

(A and C remain unchanged). Thus from now on we will project the point $(0, 0)$ onto our conic.

Next we examine whether the above transformations can possibly cause heavy losses of accuracy. They can be expressed in the matrix form as

$$\mathbf{M} \mapsto \mathbf{L}^T \tilde{\mathbf{Q}}^T \mathbf{M} \tilde{\mathbf{Q}} \mathbf{L}$$

where

$$\tilde{\mathbf{Q}} = \begin{bmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}, \qquad \mathbf{L} = \begin{bmatrix} 1 & 0 & u \\ 0 & 1 & v \\ 0 & 0 & 1 \end{bmatrix}$$

and $\mathbf{0}$ denotes a zero 2-vector. The multiplication by an orthogonal matrix $\tilde{\mathbf{Q}}$ is a numerically stable operation. The other matrix, $\mathbf{L}$, cannot have a large condition number because u and v take moderate values (of order 1). For example, if $|u| \le 1$ and $|v| \le 1$, then $\text{cond}(\mathbf{L}) \le 2 + \sqrt{3}$. So our transformations are numerically stable.

Next, if necessary, by swapping x and y we can make $|A| \le |C|$ and by negating all the signs in (32) we can make $C \ge 0$. Then we can assume that $C > 0$, as otherwise $A = C = 0$ and the conic is just a line in which case the projection procedure is trivial. Lastly, by negating the variables ($x \mapsto -x$ and $y \mapsto -y$), if necessary, we can make $D \ge 0$ and $E \ge 0$.

The orthogonality condition (3) now reads

$$-x = t(Ax + D) \quad \text{and} \quad -y = t(Cy + E) \tag{33}$$

for some real t. Solving for x and y gives

$$x = \frac{-Dt}{At + 1} \quad \text{and} \quad y = \frac{-Et}{Ct + 1}. \tag{34}$$

Substitution of these expressions into (32) gives a function

$$\mathcal{P}(t) = -D^2 t \frac{At + 2}{(At + 1)^2} - E^2 t \frac{Ct + 2}{(Ct + 1)^2} + F \tag{35}$$

whose root we need to find, because (x, y) must lie on the conic.

Next we consider three cases corresponding to different conic types:

Ellipses. If $A > 0$, then the conic is an ellipse. Its major axis is horizontal, its minor axis is vertical, and its center $(-D/A, -E/C)$ has non-positive coordinates. Therefore, it lies in the third quadrant of the xy plane.

If $D = 0$ or $E = 0$, the origin $(0, 0)$ lies on one of the axes, and this special case should be handled separately (as in Section 6). So we suppose that $D > 0$ and $C > 0$. Then the origin lies to the "North–East" of the center, hence it projects onto the ellipse to a point (x, y) that also lies to the "North–East" of the center, i.e.

$$x > -D/A \quad \text{and} \quad y > -E/C. \tag{36}$$

It now follows from (33) and (34) that

$$Ct + 1 = \frac{E}{Cy + E} > 0$$

hence $-1/C < t < \infty$. This also implies $At + 1 > 0$ because $A \le C$.

Next we show how to find t such that $\mathcal{P}(t) = 0$. Observe that

$$\lim_{t \to -1/C} \mathcal{P}(t) = +\infty \quad \text{and} \quad \lim_{t \to \infty} \mathcal{P}(t) = F - \frac{D^2}{A} - \frac{E^2}{C} < 0$$

and by direct differentiation of $\mathcal{P}$ one can see that

$$\mathcal{P}'(t) = -\frac{2D^2}{(At + 1)^3} - \frac{2E^2}{(Ct + 1)^3} < 0 \tag{37}$$

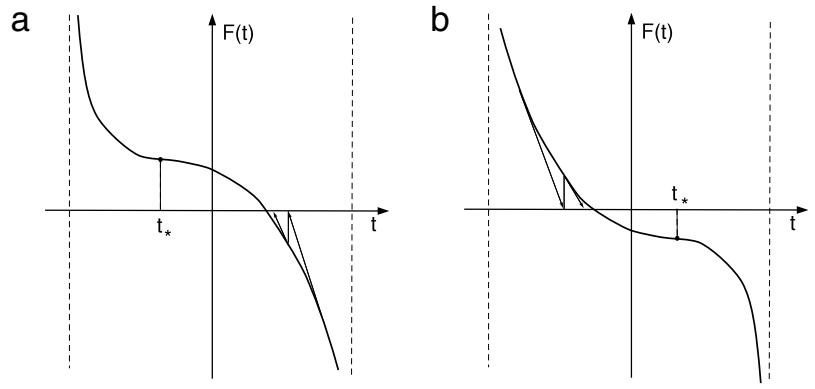

Fig. 5. Two possible appearances of $\mathcal{P}(t)$ on the interval $-1/C < t < 1/A$. Arrows show the progress of Newton's iterations toward the root.

and

$$\mathcal{P}''(t) = \frac{6AD^2}{(At+1)^4} + \frac{6CE^2}{(Ct+1)^4} > 0 \tag{38}$$

for all $t > -1/C$. Hence the function $\mathcal{P}$ is monotonically decreasing and concave, as the one shown in Fig. 3. Thus the standard Newton's method starting at any point t_0 where $\mathcal{P}(t_0) > 0$ will converge to the unique root of $\mathcal{P}$.

A starting point can be chosen as follows: if $F \geq 0$, take $t_0 = 0$ (because $\mathcal{P}(0) = F$), and if $F < 0$, then take $t_0 = \max\{t_0', t_0''\}$, where

$$t_0' = \frac{F}{D^2 - AF + D\sqrt{D^2 - AF}} \quad \text{and} \quad t_0'' = \frac{F}{E^2 - CF + E\sqrt{E^2 - CF}}. \tag{39}$$

One can easily check that $\mathcal{P}(t_0') \geq 0$ and $\mathcal{P}(t_0'') \geq 0$.

There are obvious similarities between the above procedure and the original Eberly's method. But our modification here avoids the potentially disastrous translation of the coordinate system to the ellipse's center. Also, it makes a smooth transition between the conic's types; see below.

Parabolas. If $A = 0$, then the conic is a parabola. All the above formulas and conclusions remain valid, and therefore the projection method works exactly as for ellipses, with no changes required.

By contrast, the original Eberly's method cannot be applied to parabolas as their canonical equation is different from (25). Adaptations of the original Eberly's method to parabolas and hyperbolas exist [23], but each works only for conics of a specific type. The whole procedure experiences a major disruption when the conic changes type, and the borderline cases (like nearly parabolic ellipses) cause a total breakdown.

Hyperbolas. If $A < 0$, then the conic is a hyperbola. Its major axis is still horizontal, its minor axis is still vertical, but its center $(-D/A, -E/C)$ is now in the fourth quadrant of the xy plane (where $x \geq 0$ and $y \leq 0$).

If $D = 0$ or $E = 0$, the origin $(0, 0)$ lies on one of the axes, and this special case should be handled separately. Therefore, we suppose that $D > 0$ and $C > 0$. Then the origin $(0, 0)$ lies to the "North–West" of the center, and hence it projects onto the hyperbola to a point (x, y) that also lies to the "North–West" of the center, i.e.

$$x < -D/A \quad \text{and} \quad y > -E/C. \tag{40}$$

It now follows from (33) and (34) that $At + 1 > 0$ and $Ct + 1 > 0$, i.e.,

$$-1/C < t < -1/A. \tag{41}$$

Next we turn to finding the root t of (35). Observe that

$$\lim_{t \to -1/C} \mathcal{P}(t) = +\infty \quad \text{and} \quad \lim_{t \to 1/A} \mathcal{P}(t) = -\infty$$

and $\mathcal{P}'(t) < 0$ for all t in the interval (41). Hence the function $\mathcal{P}$ is monotonically decreasing and has a unique root satisfying (41). Now $\mathcal{P}''(t)$ decreases from $+\infty$ (near $-1/C$) to $-\infty$ (near $1/A$), and it is monotonic (because $\mathcal{P}''' < 0$, as one can easily verify). Thus $\mathcal{P}$ has a unique inflection point, t_*, within the interval (41).

See Fig. 5, where two possible cases are shown: (a) the inflection point lies above the x axis, i.e., $\mathcal{P}(t_*) > 0$ and (b) the inflection point lies below the x axis. The inflection point is found by solving equation $\mathcal{P}''(t) = 0$, hence

$$t_* = -\frac{U - V}{CU - AV},$$

where

$$U = \sqrt[4]{-AD^2} \quad \text{and} \quad V = \sqrt[4]{CE^2}.$$

The value $\mathcal{P}_* = \mathcal{P}(t_*)$ at the inflection point is

$$\mathcal{P}_* = \frac{(U-V)}{(C-A)^2}\left[\frac{D^2(2C-A-AV/U)}{U} - \frac{E^2(2A-C-CU/V)}{V}\right] + F.$$

Now by computing $\mathcal{P}_*$ we can determine which case, (a) or (b), we have at hand. The standard Newton's method will converge to the root of $\mathcal{P}(t) = 0$, but the starting point t_0 must be selected wisely. In case (a) we need to choose t_0 such that $\mathcal{P}(t_0) < 0$, and in case (b) we need $\mathcal{P}(t_0) > 0$.

If $\mathcal{P}_* > 0$ and $F \le 0$ we set $t_0 = 0$ (because $\mathcal{P}(0) = F$). If $\mathcal{P}_* > 0$ and $F > 0$ we set $t_0 = \min\{t_0', t_0''\}$, cf. (39). If $\mathcal{P}_* < 0$ and $F \ge 0$ we set $t_0 = 0$. If $\mathcal{P}_* < 0$ and $F < 0$ we set $t_0 = \max\{t_0', t_0''\}$, as in the case of ellipses above. Lastly, if $\mathcal{P}_* = 0$ then $t = t_*$ is the desired root of $\mathcal{P}$.

We note that the case $\mathcal{P}_* < 0$ is handled here exactly as the cases of ellipses and parabolas above. Thus the same procedure applies to all the three types of conics—ellipses, parabolas, and hyperbolas (in the last case as long as $\mathcal{P}_* < 0$). Only for hyperbolas with $\mathcal{P}_* > 0$ does it have to be modified slightly (t_0 has to be chosen differently). The borderline case $\mathcal{P}_* = 0$ is actually the simplest one: we can skip the choice of t_0 and the Newton's iterations altogether because the root $t = t_*$ of $\mathcal{P}$ is given "for free".

Thus there are only two cases in the modified Eberly's algorithm for conics of all types, and the transition from one case to the other is smooth and painless.

Our projection problem is mathematically equivalent to the minimization of the quadratic function $P(x, y) = (x-u)^2 + (y-v)^2$ under the quadratic constraint (2). Moré [43] studied such a problem in a general form, as the minimization of an arbitrary quadratic function in $\mathbb{R}^n$ under a quadratic constraint. He introduced a parameter t, as we did in (3), and proved that the minimization problem always reduces to solving an equation $\phi(t) = 0$ in t on a certain interval $t_{\min} < t < t_{\max}$, where ϕ is a monotonic function with a unique root. He proved that there are generally two cases: one (corresponding to our ellipses) in which the function ϕ is convex or concave, whereby Moré recommends Newton's method for the minimization. In the other general case (corresponding to our hyperbolas) ϕ is neither convex or concave, whereby Moré admits that "extra care is needed" [43, p. 203], but offers no specific recommendations. In the latter case, we were successful in finding a subinterval in $(t_{\min}, t_{\max})$ where ϕ is either convex or concave, guaranteeing that an initial guess t_0 chosen from this subinterval results in the convergence of Newton's iterations. Thus our algorithm can be regarded as a practical implementation of the general recommendations of Moré.

8. Numerical tests

To test the projection methods, we generate conics (2) and points (u, v) randomly. A point (u, v) is generated using the uniform distribution in the square $|u| \le 1$, $|v| \le 1$; this ensures that u and v take values of order one.

Conics are generated in two ways. The first is a totally random selection: the parameters A, B, C, D, E, F of the conic (2) are generated using the standard normal distribution $N(0, 1)$ and then scaled so that $A^2 + B^2 + C^2 + D^2 + E^2 + F^2 = 1$. We only keep real non-degenerate conics (about 3% of the randomly generated random conics happen to be imaginary or degenerate; those are discarded).

We also discard conics that fail to cross the square $|u| \le 1$, $|v| \le 1$ containing the generated points. Our rationale is that such conics should never be used by fitting procedures—they are too far from the data points and should be rejected immediately, without precise calculation of projection footpoints and distances.

The second way of generating conics focuses on difficult, ill-conditioned types. We generate A, B, D, E, F as above and then compute $C = B^2/A + \varepsilon Z$, where Z is a standard normal random variable and ε is a very small constant. For $\varepsilon = 0$ the conic is a parabola, so for $\varepsilon \approx 0$ we get nearly parabolic conics (ellipses and hyperbolas with very large axes as described in the end of Section 6). We have set ε to 10^{-12}, 10^{-13}, 10^{-14} and found that the results changed very little, so we only report our results for $\varepsilon = 10^{-13}$.

The numerical error committed during the computation of the projected point (x, y) in our tests is measured by

$$E(x, y) = |Q(x, y)| + |R(x, y)|,$$

where Q and R are defined in (2) and (7), respectively; cf. (10). Recall the ideal projected point satisfies $Q(x, y) = R(x, y) = 0$, i.e., ideally $E(x, y) = 0$.

We note that the value of $E(x, y)$ measures numerical errors only locally, in the vicinity of the true projected point. Globally it may be misleading as it vanishes at every footpoint of the orthogonal projection, not only at the nearest one (and there are up to four such points). We have checked, separately (as described in [16]), that the computed projection point in our tests was always near the right one, so $E(x, y)$ was a proper measure of error.

Generally, however, if the point (u, v) happens to lie near the center or near the major axis of the ellipse, then determining the correct projection point becomes a delicate matter—small round off errors can easily force the projection to go to the wrong side of the ellipse. This is a common problem in geometric analysis, in which case an extra precision computer arithmetic may be needed to ensure the correct result [44, p. 6]. However, in the fitting problems that motivate our work, this is not an issue, as our main goal is to minimize the distances, and the wrong projection point would have nearly the same distance to (u, v) as the right one.

Table 1
The distribution of bad cases over $k = 1, \ldots, 13$ for four projection methods: WEP, our modified Eberly (ME), the original Eberly (OE), and the quartic equation (E-4). Here conics are generated totally randomly.

	1	2	3	4	5	6	7	8	9	10	11	12	13
WEP	0	0	0	0	0	0	0	0	0	0	0	0	53
ME	0	0	0	0	0	0	0	33	239	1909	2E4	1E5	8E5
OE	0	0	0	0	0	0	2	51	536	4911	4E4	4E5	3E6
E-4	7	11	41	114	354	1174	3743	1E4	4E4	1E5	4E5	1E6	4E6

Note: large numbers are abbreviated so that 2E4 = 20 000, 4E5 = 400 000, etc.

Table 2
The distribution of bad cases over $k = 1, \ldots, 13$ for three projection methods: WEP, our modified Eberly (ME), and the quartic equation (E-4). Here conics are ill-conditioned (nearly parabolic) with $\varepsilon = 10^{-13}$. The original Eberly's method is not included here as it crashes completely.

	1	2	3	4	5	6	7	8	9	10	11	12	13
WEP	0	0	0	0	0	0	0	0	0	0	0	17	1895
ME	0	0	0	0	0	0	0	13	124	1146	1E4	1E5	8E5
E-4	3	9	21	72	221	725	2371	1E4	2E4	7E4	2E5	8E5	3E6

Note: large numbers are abbreviated so that 2E4 = 20 000, 8E5 = 800 000, etc.

Now our goal is to investigate the numerical stability of the projection methods, so we focus *not* on average errors but rather on "worst cases", i.e., on large errors—to see how large they are and how often they occur.

The largest values of $E(x, y)$ observed in our tests were of order 0.1. So for each observed value of $E(x, y)$ we compute $k = [-\log_{10} E(x, y)]$, which is the number of zeros after the decimal point in the digital representation of $E(x, y)$. This can be roughly interpreted as the number of correct (trustworthy) decimal digits in the computed coordinates x and y.

Whenever $k \leq 13$ we regard it as a bad case and record it. In the end, we have a certain number of recorded bad cases for each $k = 1, \ldots, 13$ for each projection method. Our tables summarize the numbers of bad cases for all $k \leq 13$ and for all tested projection algorithms.

The total number of randomly generated conics and points in our tests was 10^8. Our tables only show bad cases, and one can see that their total number is relatively small for each method (even for the least accurate quartic equation method, it is less than 6×10^6, i.e., less than 6%). But these cases (their number and their distribution over k) are exactly what characterize the numerical stability of each algorithm.

Tables 1 and 2 report the results of our tests (done in MATLAB using double precision). We used matrix library functions whenever possible. Even the quartic equation (6) was solved by MATLAB function *roots* that actually computes the eigenvalues of the companion matrix; see MATLAB documentation.

We also conducted similar tests in C++ using open access matrix library *eigen3*; see [45]. In double precision, the results were almost identical to those in Tables 1 and 2. With long double precision (allowing for up to 19–20 accurate decimal digits), the number of accurate digits generally increased by three, i.e., all the distributions presented in Tables 1 and 2 were roughly shifted three positions to the right (for example, the first bad case for the WEP method on totally random conics appeared in column 16, rather than column 13, etc.). Our code is posted on the web [41].

The WEP method is clearly superior to others, while the quartic equation algorithm is the most vulnerable. The original Eberly's method performs well in typical cases (though it is still far behind the WEP), but in nearly singular cases it breaks down completely, for the reasons explained in Section 6. The modified Eberly's method remains quite stable.

Interestingly, the modified Eberly's method and the quartic equation method do not deteriorate at all in nearly singular situations (in fact, the number of bad cases seems to decrease, strangely enough—compare Tables 1 and 2). But the WEP method is still far better than the other two.

One can improve the performance of all these methods by "polishing" the projected points with Newton's iterations (Section 3). Just two iterations are enough to make all the results nearly perfect—the smallest value of $k = [-\log_{10} E(x, y)]$ will then be 14 for all 10^8 randomly generated conics and points (u, v) and for all our methods presented in Tables 1 and 2, except E-4. For the quartic equation method, a few bad cases (about 10–15, out of 10^8) cannot be fixed by Newton's iterations, they are irreparable. This fact underscores the deficiency of the E-4 method.

9. Conclusions and extensions to 3D

There are several algorithms for projecting 2D points onto conics that are mathematically proven to give (or converge to) correct results. However, when it comes to numerical computations, some of these methods turn out to be intrinsically vulnerable and prone to large computational errors; hence they should be avoided in practice.

There remain two competing algorithms that are both theoretically and numerically reliable—the Wijewickrema–Esson–Papliński (WEP) method and the modified Eberly's (ME) method. The former is more complicated and harder to program, but its precision is virtually perfect in all cases. The latter is simpler and easier to code, but as a stand-alone procedure it is less precise than the WEP. By combining the ME with a couple of Newton's iterations one can achieve the same accuracy as that of the WEP, but the procedure gets somewhat more complicated.

The WEP method is flexible enough to allow various implementations, with and without matrix library functions. The use of the latter makes coding easier, but the execution time in C++ gets 5–10 times longer. If one avoids matrix library functions and optimizes the WEP for speed, it will be only 1.5 times slower than ME (again, measured in C++).

We emphasize that the two best algorithms, the WEP and ME, are practically brand new. The WEP was presented at a recent conference [26] but never published in a journal. The ME is proposed here for the first time.

Our next goal is to extend these studies to 3D, i.e., to investigate the projection of spacial points (x, y, z) on quadratic surfaces (quadrics). Here we only touch upon the arising difficulties and new challenges.

First, there will be no analogue of the quartic equation method, as the 3D problem reduces to a polynomial equation of degree *six*, rather than four. Thus there will be no analytic solution along the lines of Section 2.

The WEP can be extended to 3D, in a sense, but instead of the intersection of two conics one has to deal with the intersection of three quadrics, a much more complicated task; see some remarks in [26]. A detailed implementation of this method is yet to be developed and its efficiency and accuracy are yet to be investigated.

The original Eberly's method extends to 3D, see [23], but for the same reasons as in 2D it is too vulnerable. A modification of Eberly's method in 3D (along the lines of Section 7) is yet to be designed. This is all work in progress and we hope to report the results in a separate paper.

Acknowledgments

We thank A. Korepanov for helpful discussions and his assistance in using C++ matrix library *eigen3*. N.C. was partially supported by National Science Foundation, grant DMS-0969187.

References

[1] R.O. Duda, P.E. Hart, The use of Hough transform to detect lines and curves in pictures, Commun. ACM 15 (1972) 11–15.
[2] T. Hastie, W. Stuetzle, Principal curves, J. Amer. Statist. Assoc. 84 (1989) 502–516.
[3] Z. Dreznera, S. Steinerb, G.O. Wesolowsky, On the circle closest to a set of points, Comput. Oper. Res. 29 (2002) 637–650.
[4] R.J. Adcock, Note on the method of least squares, Analyst 4 (1877) 183–184.
[5] C.H. Kummell, Reduction of observation equations which contain more than one observed quantity, Analyst 6 (1879) 97–105.
[6] N. Chernov, Circular and Linear Regression: Fitting Circles and Lines by Least Squares, in: Chapman & Hall/CRC Monographs on Statistics and Applied Probability, vol. 117, 2010.
[7] S.M. Robinson, Fitting spheres by the method of least squares, Commun. ACM 4 (1961) 491.
[8] A. Thom, A statistical examination of the megalithic sites in Britain, J. Roy. Statist. Soc. Ser. A 118 (1955) 275–295.
[9] A. Albano, Representation of digitized contours in terms of conic arcs and straight-line segments, Comput. Graph. Image Process. 3 (1974) 23–33.
[10] R.H. Biggerstaff, Three variations in dental arch form estimated by a quadratic equation, J. Dent. Res. 51 (1972) 1509.
[11] F.L. Bookstein, Fitting conic sections to scattered data, Comput. Graph. Image Process. 9 (1979) 56–71.
[12] K. Paton, Conic sections in chromosome analysis, Pattern Recognit. 2 (1970) 39–51.
[13] V. Pratt, Direct least-squares fitting of algebraic surfaces, Comput. Graph. 21 (1987) 145–152.
[14] P.D. Sampson, Fitting conic sections to very scattered data: an iterative refinement of the Bookstein algorithm, Comput. Graph. Image Process. 18 (1982) 97–108.
[15] G. Taubin, Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations, with applications to edge and range image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 13 (1991) 1115–1138.
[16] S.J. Ahn, Least Squares Orthogonal Distance Fitting of Curves and Surfaces in Space, in: LNCS, vol. 3151, Springer, Berlin, 2004.
[17] W. Gander, G.H. Golub, R. Strebel, Least squares fitting of circles and ellipses, BIT 34 (1994) 558–578.
[18] Z. Zhang, Parameter estimation techniques: a tutorial with application to conic fitting, Int. J. Image Vis. Comput. 15 (1997) 59–76.
[19] K. Kanatani, P. Rangarajan, Hyper least squares fitting of circles and ellipses, Comput. Statist. Data Anal. 55 (2011) 2197–2208.
[20] S.J. Ahn, W. Rauh, H.J. Warnecke, Least-squares orthogonal distances fitting of circle, sphere, ellipse, hyperbola, and parabola, Pattern Recognit. 34 (2001) 2283–2303.
[21] S.J. Ahn, W. Rauh, M. Recknagel, Least squares orthogonal distances fitting of implicit curves and surfaces, in: LNCS, vol. 2191, 2001, pp. 398–405.
[22] S.J. Ahn, W. Rauh, H.S. Cho, Orthogonal distances fitting of implicit curves and surfaces, IEEE Trans. Pattern Anal. Mach. Intell. 24 (2002) 620–638.
[23] N. Chernov, H. Ma, Least squares fitting of quadratic curves and surfaces, in: S.R. Yoshida (Ed.), Computer Vision, Nova Science Publishers, 2011, pp. 285–302.
[24] M. Aigner, B. Jüttler, Gauss–Newton type techniques for robustly fitting implicitly defined curves and surfaces to unorganized data points, in: Shape Modeling International, 2008, pp. 121–130.
[25] P. Sturm, P. Gargallo, Conic fitting using the geometric distance, in: Proc. Asian Conf. Comp. Vision, Vol. 2, Tokyo, Japan, 2007, pp. 784–795.
[26] S. Wijewickrema, C. Esson, A. Papliński, Orthogonal distance least squares fitting: a novel approach, in: Proc. CVICGTA (Lisboa, Portugal, Feb. 5–8, 2009), in: Commun. Comp. Inf. Sci., vol. 68, 2010, pp. 255–268.
[27] N. Chernov, Q. Huang, H. Ma, Does the best fitting curve always exist? ISRN Probab. Statist. (2012) Article ID 895178.
[28] 3D Game Engine Design, 2nd ed., Morgan Kaufmann Publishers, San Francisco, CA, 2007, See also Internet article Distance from a point to an ellipse in 2D, Geometric Tools, LLC. www.geometrictools.com.
[29] M. Aigner, B. Jüttler, Robust computation of foot points on implicitly defined curves, in: M. Daehlen, et al. (Eds.), Mathematical Methods for Curves and Surfaces, Tromso 2004, Nashboro Press, 2005, pp. 1–10.
[30] L. Trefethen, D. Bau III, Numerical Linear Algebra, SIAM, 1997.
[31] T. Strong, Elementary and Higher Algebra, Pratt and Oakley, 1859.
[32] B. Christianson, Solving Quartics using Palindromes, Math. Gaz. 75 (1991) 327–328.
[33] H.W. Turnbull, Theory of Equations, fourth ed., Oliver and Boyd, London, 1947.
[34] S. Neumark, Solution of Cubic and Quartic Equations, Pergamon Press, Oxford, 1965.
[35] M.D. Yacoub, G. Fraidenraich, A new simple solution of the general quartic equation, Math. Gaz. 2011 (in press).
[36] D. Herbison-Evans, Solving quartics and cubics for graphics, Tech. Report TR94-487, Basser Dept. Comp. Sci., Univ. of Sydney, Australia.
[37] S. Shmakov, A universal method of solving quartic equations, Int. J. Pure Appl. Math. 71 (2011) 251–259.
[38] J.G. Semple, G.T. Kneebone, Algebraic Projective Geometry, Oxford U. Press, Oxford, 1956.
[39] W. Kahan, To solve a real cubic equation (lecture notes for a numerical analysis course), Tech. Rep. UCB/CPAM-86-352, Center for Pure and Applied Mathematics, University of California, Berkeley, CA, 10 November 1986, Access Number ADA206859. www.dtic.mil/dtic/tr/fulltext/u2/a206859.pdf.

[40] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes. The Art of Scientific Computing, third ed., Cambridge University Press, 2007, Earlier editions available free on-line.
[41] http://www.math.uab.edu/~chernov/cl.
[42] Y. Nievergelt, Fitting conics of specific types to data, Linear Algebra Appl. 378 (2004) 1–30.
[43] J.J. Moré, Generalizations of the trust region problem, Optim. Methods Softw. 2 (1993) 189–209.
[44] W. Kahan, On the cost of floating-point computation without extra-precise arithmetic, http://www.cs.berkeley.edu/~wkahan/Qdrtcs.pdf, 20 November 2004.
[45] http://eigen.tuxfamily.org.