Least squares fitting of circles

N. Chernov and C. Lesort
Department of Mathematics
University of Alabama at Birmingham
Birmingham, AL 35294, USA

November 18, 2008

Abstract

Fitting standard shapes or curves to incomplete data (which represent only a small part of the curve) is a notoriously difficult problem. Even if the curve is quite simple, such as an ellipse or a circle, it is hard to reconstruct it from noisy data sampled along a short arc. Here we study the least squares fit (LSF) of circular arcs to incomplete scattered data. We analyze theoretical aspects of the problem and reveal the cause of unstable behavior of conventional algorithms. We also find a remedy that allows us to build another algorithm that accurately fits circles to data sampled along arbitrarily short arcs.

1 Introduction

Fitting simple contours (primitives) to experimental data is one of the basic problems in pattern recognition and computer vision. The simplest contours are linear segments and circular arcs. The need of approximating scattered points by a circle or a circular arc also arises in physics [8, 12, 18], biology and medicine [4, 25, 26], archeology [14], industry [13, 19, 34], etc. This problem was studied since at least early sixties [28], and by now has a long history.

A variety of good algorithms are available for fitting circles to data sampled along a full circle or a sufficiently large arc, we discuss the best ones below. However, if data are sampled along a small arc ($20^{\rm o}$ or $10^{\rm o}$ or less), many known algorithms develop various singularities and become unstable. This problem plagues the experimenters in many areas. One of them is high energy physics, where planar images of the tracks of elementary particles are circular arcs whose radius is proportional to the energy of the particle. Fast particles, common in modern experiments, leave a trace that is barely curved, but an accurate estimate of the curvature is essential for the computation of the particle's energy.

We attempt here a theoretical analysis of the least squares fitting problem (though it is restricted to circular arcs, it can be applied to more general curves as well, and we plan to extend it to conics in a separate paper). Our study reveals the causes of the trouble experienced by conventional fitting algorithms when applied to incomplete data (representing small arcs). We combine elements of the existing algorithms to build a new one that remains stable and accurate when applied to arbitrary short arcs. Our conclusions are supported by numerical experiments with simulated data.

2 Theoretical aspects

The least squares fit (LSF) is based on minimizing the mean square distance from the fitting curve to data points. Given n points (x_i, y_i), $1 \le i \le n$, the objective function is defined by

$$\mathcal{F} = \sum_{i=1}^{n} d_i^2 \tag{2.1}$$

where d_i is the Euclidean (geometric) distance from the point (x_i, y_i) to the curve. When fitting circles, one parametrizes those by the equation

$$(x-a)^2 + (y-b)^2 = R^2 \tag{2.2}$$

where (a, b) is the center and R the radius, and then

$$d_i = \sqrt{(x_i-a)^2 + (y_i-b)^2} - R \tag{2.3}$$

The LSF is equivalent [3] to the maximum likelihood estimation under the common assumption that the noise has gaussian distribution [3, 6, 8, 15, 17, 37], hence the LSF is considered as a statistically optimal method, see a more detailed account in [10]. Unfortunately, there is no direct algorithm for the least squares fitting of curves (other than straight lines), the minimization of (2.1) is a nonlinear problem that has no closed form solution. All known algorithms are either iterative and costly or approximative by nature.

We begin our study of the problem (2.1)–(2.3) by discussing its theoretical aspects, such as the existence and uniqueness of its solution, the right choice of parameters to work with, and the general behavior of the objective function $\mathcal{F}$ on the parameter space. These issues are rarely discussed in the literature, but they are essential for the understanding of advantages and disadvantages of practical algorithms.

Our first questions are the existence and uniqueness of a solution of (2.1)–(2.3): Does the function $\mathcal{F}$ attain its minimum? Assuming that it does, is the minimum unique?

2.1 Existence of LSF. The function $\mathcal{F}$ is obviously continuous in the circle parameters a, b, R. According to a general principle, a continuous function always attains a minimum (possibly, not unique) if it is defined on a closed and bounded (i.e., compact) domain.

Our function $\mathcal{F}$ is defined for all a, b and all $R \geq 0$, so its domain is not compact, and for this reason the function $\mathcal{F}$ fails to attain its minimum in some cases.

For example, let $n \geq 3$ distinct points lie on a straight line. Then one can approximate the data by a circle arbitrarily well and make $\mathcal{F}$ arbitrarily close to zero, but since no circle can interpolate $n \geq 3$ collinear points, we will always have $\mathcal{F} > 0$. Hence, the least squares fit by circles is, technically, impossible. For any circle one can find another circle that fits the data better. The best fit here is given by the straight line trough the data points, which yields $\mathcal{F} = 0$. If we want the LSF to exist, we have to allow lines, as well as circles, in our model, and from now on we do this.

One can prove now that the function $\mathcal{F}$ defined on circles *and* lines always attains its minimum, for any set of $n \geq 1$ points, and so the existence of the LSF is guaranteed.

We should note that if the data points are generated randomly with a continuous probability distribution (such as normal or uniform), then the probability that the LSF returns a line, rather than a circle, is zero. On the other hand, if the coordinates of the data points are discrete (such as pixels on a computer screen), then lines may appear with a positive probability and have to be reckoned with.

2.2 Uniqueness of LSF. Surprisingly, the LSF is not unique. Even if $\mathcal{F}$ takes its minimum on a circle (rather than a line), the best fitting circle may not be unique, as several other circles may minimize $\mathcal{F}$ as well. We could not find such examples in the literature, so we provide our own here.

Examples of multiple minima. Let four data points $(\pm 1, 0)$ and $(0, \pm 1)$ make a square centered at the origin. We place another $k \geq 4$ points identically at the origin $(0, 0)$ to have a total of $n = k + 4$ points. This configuration is invariant under a rotation through the right angle about the origin. Hence, if a circle minimizes $\mathcal{F}$, then by rotating that circle through $\pi/2$, π, and $3\pi/2$ we get three other circles that minimize $\mathcal{F}$ as well.

Thus, we get four distinct best fitting circles, unless either (i) the best circle is centered at the origin and hence is invariant under the rotations itself or (ii) the best fit is a line. So we need to rule out (i) and (ii). This involves some elementary calculations. If a circle has radius r and center at $(0, 0)$, then $\mathcal{F} = 4(1 - r^2) + kr^2$. The minimum of this function is attained at $r = 4/(k + 4)$ and is equal to $\mathcal{F}_0 = 4k/(k + 4)$. Also, if the best fit was a line, then such a line would pass through the origin and would give $\mathcal{F}_1 = 2$. On the other hand, consider the circle passing through the three points $(0, 0)$, $(0, 1)$, and $(1, 0)$. It only misses two other points, $(-1, 0)$ and $(0, -1)$, and it is easy to see that it gives $\mathcal{F} < 2$, which is less than $\mathcal{F}_1$ and $\mathcal{F}_0$ whenever $k \geq 4$. Hence, the best fit here is a circle, rather than a line, and the best circle is not centered at the origin, hence our argument applies.

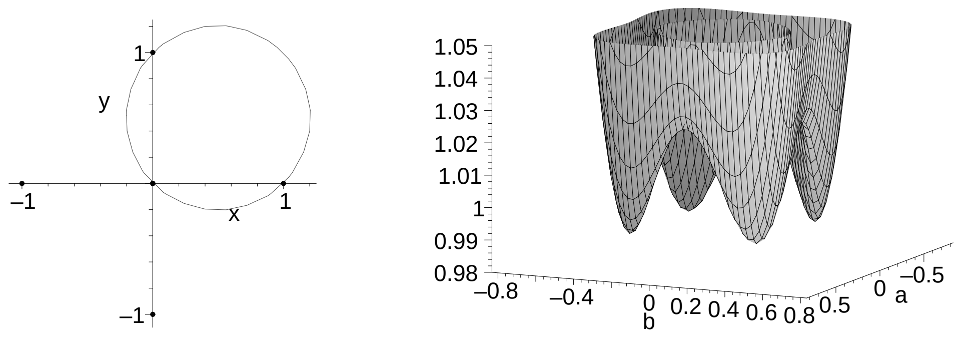


Figure 1: A data set (left) for which the objective function (right) has four minima.

Figure 1 illustrates our example, it gives the plot of $\mathcal{F}$ with four distinct minima (the plotting method is explained below). By changing the square to a rectangle one can make $\mathcal{F}$ have exactly two minima. By replacing the square with a regular m-gon and increasing the number of identical points at $(0, 0)$ one can construct $\mathcal{F}$ with exactly m minima for any $m \geq 3$, see details on our web page [9].

Of course, if the data points are generated randomly with a continuous probability distribution, then the probability that the objective function $\mathcal{F}$ has multiple minima is zero. In particular, small random perturbations of the data points in our example in Fig. 1 will slightly change the values of $\mathcal{F}$ at its minima, so that one of them will become a global (absolute) minimum and three others will be local (relative) minima.

2.3 Local minima of the objective function. Generally, local minima are undesirable, since they can trap iterative algorithms and lead to false solutions.

We have investigated how frequently the function $\mathcal{F}$ has local minima (and how many). In one experiment, n data points were generated randomly with a uniform distribution in the unit square $0 < x, y < 1$. Then we applied the Levenberg-Marquard algorithm (described below) starting at 1000 different, randomly selected initial conditions. Every point of convergence was recorded as a minimum (local or global) of $\mathcal{F}$. If there were more than one point of convergence, then one of them was the global minimum and the others were local minima. Table 1 shows the fraction of simulated samples where $\mathcal{F}$ had 0,1,2 or more local minima for $n = 5, \ldots, 100$ data points.

	5	10	15	25	50	100
0	0.879	0.843	0.883	0.935	0.967	0.979
1	0.118	0.149	0.109	0.062	0.031	0.019
≥ 2	0.003	0.008	0.008	0.003	0.002	0.002

Table 1. Frequency of appearance of 0, 1, 2 or more local minima of $\mathcal{F}$ when $n = 5, \ldots, 100$ points are generated randomly.

We see, surprisingly, that local minima only occur for less than 15% of generated samples. The more points are generated, the less frequently the function $\mathcal{F}$ happens to have any local minima. Multiple local minima turn up very rarely, if at all. The maximal number of local minima we observed in any single sample was four, it happened only a few times in almost a million random samples we tested.

Generating points with a uniform distribution produces completely "chaotic" samples without any predefined pattern. This is, in a sense, the worst case scenario. When we generated points along a circle or a circular arc (with small noise), then local minima virtually never occur. For example, if $n = 10$ points are sampled along a 90° circular arc of radius $R = 1$ with a Gaussian noise at level $\sigma = 0.05$, then the frequency of appearance of local minima is as low as 0.001.

Therefore, in typical applications the objective function $\mathcal{F}$ is very likely to have a unique (global) minimum and no local minima. Does this mean that a standard iterative algorithm, such as the steepest descent or the Nelder-Mead simplex or the Gauss-Newton or the Levenberg-Marquardt, would converge to the global minimum from any starting point? Unfortunately, this is not the case, as we demonstrate next.

2.4 Plateaus and valleys on the graph of the objective function. In order to examine the behavior of $\mathcal{F}$ we found a way to visualize its graph. Even though $\mathcal{F}(a, b, R)$ is a function of three parameters, it happens to be just a quadratic polynomial in R, when a and b are fixed. So it has a unique global minimum in R that can be easily found. If we denote $r_i = \sqrt{(x_i - a)^2 + (y_i - b)^2}$, then the minimum of $\mathcal{F}$ with respect to R is attained at

$$\hat{R} = \bar{r} := \frac{1}{n} \sum_{i=1}^{n} r_i \tag{2.4}$$

This allows us to eliminate R and express $\mathcal{F}$ as a function of a, b by $\mathcal{F} = \sum_{i=1}^{n} (r_i - \bar{r})^2$. A function of two variables can be easily plotted. This is exactly how we did it in Fig. 1.

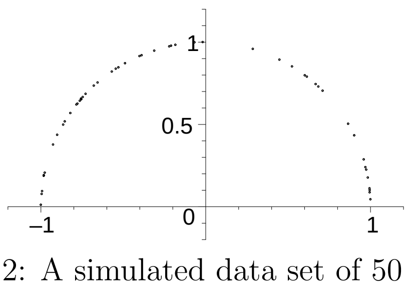
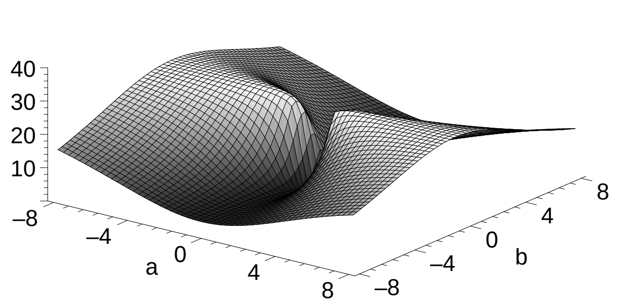


Figure 2: A simulated data set of 50 points.

Now, Fig. 2 presents a typical random sample of $n = 50$ points generated along a circular arc (the upper half of the unit circle $x^2 + y^2 = 1$) with a Gaussian noise added at level $\sigma = 0.01$. Fig. 3 shows the graph of $\mathcal{F}$ plotted by MAPLE in two different scales. One can clearly see that $\mathcal{F}$ has a global minimum around $a = b = 0$ and no local minima. Fig. 4 presents a flat grey scale contour map, where darker colors correspond to deeper parts of the graph (smaller values of $\mathcal{F}$).



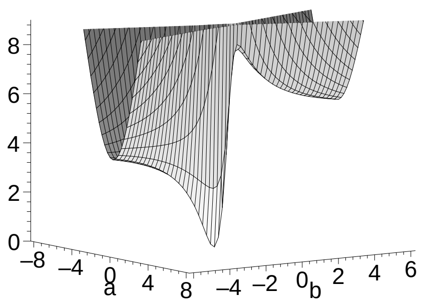


Figure 3: The objective function $\mathcal{F}$ for the data set shown in Fig. 2: a large view (left) and a vicinity of the minimum (right).

Fig. 3 shows that the function $\mathcal{F}$ does not grow as $a, b \to \infty$. In fact, it is bounded, i.e. $\mathcal{F}(a, b) \le \mathcal{F}_{\max} < \infty$. The boundedness of $\mathcal{F}$ actually explains the appearance of large nearly flat plateaus and valleys in Fig. 3 that stretch out to infinity in some directions. If an iterative algorithm starts somewhere in the middle of such a plateau or a valley or gets there by chance, it will have hard time moving at all, since the gradient of $\mathcal{F}$ will almost vanish. We indeed observed conventional algorithms getting "stuck" on flat plateaus or valleys in our tests. We will also describe an algorithm that does not have this drawback.

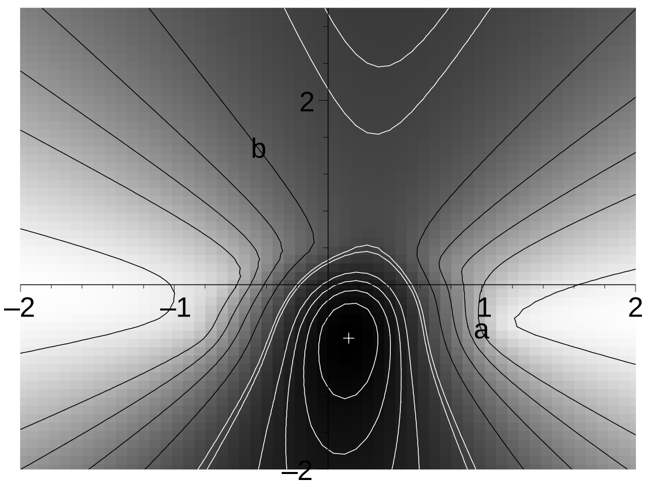


Figure 4: A grey-scale contour map of the objective function $\mathcal{F}$. Darker colors correspond to smaller values of $\mathcal{F}$. The minimum is marked by a cross.

Second, there are two particularly interesting valleys that stretch roughly along the line $a = 0$ on Figs. 3 and 4. One of them, corresponding to $b < 0$, has its bottom point at the minimum of $\mathcal{F}$. The function $\mathcal{F}$ slowly decreases along that valley as it approaches the minimum. Hence, any iterative algorithm starting in that valley or getting there by chance should, ideally, find its way downhill and arrive at the minimum of $\mathcal{F}$.

The other valley corresponds to $b > 0$, it is separated from the global minimum of $\mathcal{F}$ by a ridge. The function $\mathcal{F}$ slowly decreases along this valley as b grows. Hence, any iterative algorithm starting in this valley or getting there "by accident" will be forced to move up along the b axis, away from the minimum of $\mathcal{F}$, and escape to infinity.

If an iterative algorithm starts at a randomly chosen point, it may go down into either valley, and there is a good chance that it descends into the second (wrong) valley and then diverges. For the sample in Fig. 2, we applied the Levenberg-Marquardt algorithm starting at a randomly selected point in the square 5×5 about the centroid $x_c = \frac{1}{n}\sum x_i$, $y_c = \frac{1}{n}\sum y_i$ of the data. We found that the algorithm escaped to infinity with probability about 50%.

Unfortunately, such "escape valleys" are inevitable. We have proved [9] that for typical every data samples (i.e. with probability one) there was a pair of valleys stretching out in opposite directions, so that one valley descends to the minimum of $\mathcal{F}$, while the other valley descends toward infinity. This is the reason why iterative algorithms often fail to fit a circular arc to incomplete data.

2.5 Choice of parametrization. The trouble with the natural circle parameters a, b, R is that they become arbitrarily large when the data are incomplete and have to be approximated by a circular arc with low curvature. Not only does this lead to a catastrophic loss of accuracy when two large nearly equal quantities are subtracted in (2.3), but this is ultimately responsible for the appearance of flat plateaus and descending valleys that cause failures of iterative algorithms.

We now adopt an alternative parametrization used in [27, 15, 37], in which the equation of a circle is

$$A(x^2 + y^2) + Bx + Cy + D = 0 \tag{2.5}$$

Note that this gives a circle when $A \neq 0$ and a line when $A = 0$, so it conveniently combines both types of contours in our work. The parameters A, B, C, D must satisfy the inequality $B^2 + C^2 - 4AD > 0$ in order to define a nontrivial curve. Since the parameters only need to be defined up to a scalar multiple, we can impose a constraint

$$B^2 + C^2 - 4AD = 1 \tag{2.6}$$

If we additionally require that $A \geq 0$, then every circle or line will correspond to a unique quadruple (A, B, C, D) and vice versa. For technical reasons, though, we do not restrict A, so that circles have a duplicate parametrization, see (2.8) below. The conversion formulas between the geometric parameters (a, b, R) and the algebraic ones are

$$a = -\frac{B}{2A}, \qquad b = -\frac{C}{2A}, \qquad R = \frac{1}{2|A|} \tag{2.7}$$

and

$$A = \pm\frac{1}{2R}, \qquad B = -2Aa, \qquad C = -2Ab, \qquad D = \frac{B^2 + C^2 - 1}{4A} \tag{2.8}$$

The distance from a point (x_i, y_i) to the circle can be expressed, after some algebraic manipulations, as

$$d_i = 2\,\frac{P_i}{1+\sqrt{1+4AP_i}} \tag{2.9}$$

where

$$P_i = A(x_i^2 + y_i^2) + Bx_i + Cy_i + D \tag{2.10}$$

One can check that $1 + 4AP_i \geq 0$ for all i, see below, so that the function (2.9) is well defined.

The parametrization by A, B, C, D has several advantages. First, there is no need to work with arbitrarily large values of parameters anymore. One can effectively reduce the parameter space to a box

$$\{|A| < A_{\max}, |B| < B_{\max}, |C| < C_{\max}, |D| < D_{\max}\} \tag{2.11}$$

where $A_{\max}$, $B_{\max}$, $C_{\max}$, $D_{\max}$ can be determined explicitly [9]. This conclusion is based on some technical analysis, and we only outline its main ideas. Given a sample (x_i, y_i), $1 \leq i \leq n$, let $d_{\max} = \max_{i,j} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ denote the maximal distance between the data points. Then we observe that (i) the distance from the best fitting line or circle to the centroid of the data (x_c, y_c) does not exceed $d_{\max}$. It also follows from (2.4) that (ii) the best fitting circle has radius $R \geq d_{\max}/n$, hence $|A| \leq n/2d_{\max}$. Thus the model can be restricted to circles and lines satisfying the above conditions (i) and (ii). Under these conditions, the parameters A, B, C, D are bounded by some constants $A_{\max}$, $B_{\max}$, $C_{\max}$, $D_{\max}$. A detailed proof of this fact is contained on our web page [9].

Second, the objective function is now smooth in A, B, C, D on the entire parameter space. In particular, no singularities occur as a circular arc approaches a line. Circular arcs with low curvature correspond to small values of A, lines correspond to $A = 0$, and the objective function and all its derivatives are well defined at $A = 0$.

Third, recall the two valleys shown on Figs. 3-4, which we have proved to exist for almost every data sample [9]. In the new parameter space they merge and become the two halves of one valley stretching continuously across the hyperplane $A = 0$ and descending to the minimum of $\mathcal{F}$. Therefore, any iterative algorithm starting *anywhere* in that (now unique) valley would converge to the minimum of $\mathcal{F}$ (maybe crossing the plane $A = 0$ on its way). There is no escape to infinity anymore.

3 Geometric fit

3.1 Three popular algorithms. The minimization of the nonlinear function $\mathcal{F}$ cannot be accomplished by a finite algorithm. Various iterative algorithms have been applied to this end. The most successful and popular are

(a) the Levenberg-Marquardt method.

(b) Landau algorithm [20].

(c) Späth algorithm [31, 32]

Here (a) is a short name for the classical Gauss-Newton method with the Levenberg-Marquardt correction [22, 23]. It can effectively solve any least squares problem of type (2.1) provided the first derivatives of d_i's with respect to the parameters can be computed. The Levenberg-Marquardt algorithm is quite stable and reliable, and it usually converges rapidly (if the data points are close to the fitting contour, the convergence is nearly quadratic). Fitting circles with the Levenberg-Marquardt method is described in many papers, see, e.g. [30].

The other two methods are circle-specific. The Landau algorithm employs a simple fixed-point iterative scheme, nonetheless it shows a remarkable stability and is widely used in practice. Its convergence, though, is linear [20]. The Späth algorithm makes a clever use of an additional set of (dummy) parameters and is based on alternating minimization with respect to the dummy parameters and then with respect to the main parameters (a, b, R). At each step, one set of parameters is fixed, and a *global* minimum of the objective function $\mathcal{F}$ with respect to the other parameter set is found, which guarantees (at least theoretically) that $\mathcal{F}$ decreases at every iteration. The convergence of Späth's algorithm is, however, known to be slow [2].

The cost per iteration is about the same for all the three algorithms: the Levenberg-Marquardt requires $12n + 41$ flops per iteration, Landau takes $11n + 5$ flops per iteration and for Späth the flop count is $11n + 13$ per iteration (in all cases, a prior centering of the data is assumed, i.e. $x_c = y_c = 0$).

We have tested the performance of these three algorithms experimentally, and the results are reported in Section 3.3.

3.2 A nonconventional algorithm for circle fitting. In Section 2.5 we introduced parameters A, B, C, D subject to the constraint (2.6). Here we show how the Levenberg-Marquardt scheme can be applied to minimize the function (2.1) in the parameter space (A, B, C, D).

First, we introduce a new parameter – an angular coordinate θ defined by

$$B = \sqrt{1 + 4AD}\,\cos\theta, \qquad C = \sqrt{1 + 4AD}\,\sin\theta,$$

so that θ will replace B and C. Now one can perform an unconstrained minimization of $\mathcal{F} = \sum d_i^2$ in the three-dimensional parameter space (A, D, θ). The distance d_i is expressed by (2.9) with

$$\begin{aligned} P_i &= A(x_i^2 + y_i^2) + \sqrt{1 + AD}\,(x_i \cos\theta + y_i \sin\theta) + D \\ &= Az_i + Eu_i + D \end{aligned}$$

where we denote, for brevity, $z_i = x_i^2 + y_i^2$, $E = \sqrt{1 + 4AD}$, and $u_i = x_i \cos\theta + y_i \sin\theta$. The first derivatives of d_i with respect to the parameters are

$$\partial d_i/\partial A = (z_i + 2Du_i/E)R_i - d_i^2/Q_i$$

$$\partial d_i/\partial D = (2Au_i/E + 1)R_i$$

$$\partial d_i/\partial \theta = (-x_i \sin\theta + y_i \cos\theta)ER_i$$

where $Q_i = \sqrt{1+4AP_i}$ and $R_i = 2(1 - Ad_i/Q_i)/(Q_i + 1)$. Then one can apply the standard Levenberg-Maquardt scheme. The resulting algorithm is more complicated and costly than the methods described in 3.1, it requires $39n+40$ flops and one trigonometric function call per iteration. But it converges in fewer iterations than other methods, so their overall speeds are comparable, see the next section.

This approach has some pitfalls – the function $\mathcal{F}$ is singular (its derivatives are discontinuous) when either $1+4AP_i = 0$ (for some i) or $1+4AD = 0$. To investigate these singularities we note that $1+4AP_i = [(x_i-a)^2 + (y_i-b)^2]/R^2$. This quantity vanishes if $x_i = a$ and $y_i = b$, i.e. when a data point coincides with the circle's center. This is extremely unlikely to occur, and indeed it has never happened in our tests, so that we did not use any security measures against the singularity $1+4AP_i = 0$.

On the other hand, $1+4AD = (a^2+b^2)/R^2$, which vanishes whenever $a = b = 0$. This singularity turns out to be more destructive – whenever the circle's center computed iteratively approaches the origin, the algorithm tends to stall because it persistently tries to enter the forbidden area $1+4AD < 0$. We found a simple remedy: whenever the algorithm attempts to compute A and D for which $1+4AD < 0$, we shift the origin by adding a vector (u, v) to all the data coordinates (x_i, y_i), and then recompute the parameters (A, D, θ) accordingly. The vector (u, v) should be of size comparable to the average distance between the data points, and its direction can be chosen randomly. The shift has to be applied only occasionally and its relative cost is low.

3.3 Experimental tests. We have generated 10000 samples of n points randomly with a uniform distribution in the unit square $0 < x, y < 1$. For each sample, we generated 1000 initial guesses by selecting a random circle center (a, b) in a square 5×5 around the centroid (x_c, y_c) of the sample and then computing the radius R by (2.4). Every triple (a, b, R) was then used as an initial guess, and we ran all the four algorithms described in 3.1–3.2 starting at the point (a, b, R).

For each sample and for each algorithm, we have determined the number of runs, from the 1000 random initial guesses, when the iterations converged to the global minimum of $\mathcal{F}$. Dividing that number by the total number of generated initial guesses, 1000, we obtained the *probability* of convergence to the minimum of $\mathcal{F}$. We also computed the average number of iterations in which the convergence took place. At this stage, the samples for which the function $\mathcal{F}$ had any local minima, in addition to the global minimum, were eliminated (the fraction of such samples was less than 15%, as one can see in Table 1). The remaining majority of samples were used to compute the overall characteristics of each algorithm: the grand average probability of convergence to the minimum of $\mathcal{F}$ and the grand mean number of iterations the convergence took. We note that, since samples with local minima are eliminated at this stage, the probability of convergence is a true measure of the algorithm's reliability (rather than the complexity of $\mathcal{F}$). All failures to converge is only the algorithm's fault.

This experiment was repeated for $n = 5, 10, \ldots, 100$. The results are presented in Figures 5 and 6, where the algorithms are marked as follows: LAN for Landau, SPA for Späth, LMC for the canonical Levenberg-Maquardt in the (a, b, R) parameter space, and LMA for the alternative Levenberg-Maquardt in the (A, D, θ) parameter space.

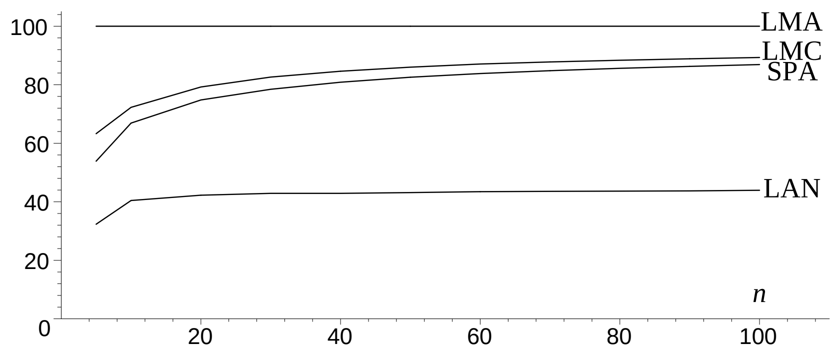


Figure 5: The probability of convergence to the minimum of $\mathcal{F}$ starting at a random initial guess.

Figure 5 shows the probability of convergence to the minimum of $\mathcal{F}$, it clearly demonstrates the superiority of the LMA method. Figure 6 presents the average cost of convergence, in terms of flops per data point, for all the four methods. The fastest algorithm is the canonical Levenberg-Marquardt (LMC). The alternative Levenberg-Marquardt (LMA) is almost twice as slow. The Späth and Landau methods happen to be far more expensive, in terms of the computational cost, than both Levenberg-Marquardt schemes.

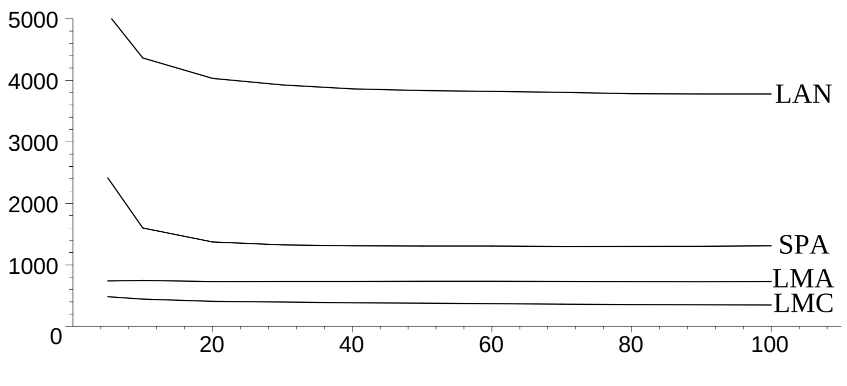


Figure 6: The average cost of computations, in flops per data point.

The poor performance of the Landau and Späth algorithms is illustrated in Fig. 7. It shows a typical path taken by each of the four procedures starting at the same initial point $(1.35, 1.34)$ and converging to the same limit point $(0.06, 0.03)$ (the global minimum of $\mathcal{F}$). Each dot represents one iteration. For LMA and LMC, subsequent iterations are connected by grey lines. One can see that LMA (hollow dots) heads straight to the target and reaches it in about 5 iterations. LMC (solid dots) makes a few jumps back and forth but arrives at the target in about 15 steps. On the contrary, SPA (square dots) and LAN (round dots) advance very slowly and tend to make many short steps as they approach the target (in this example SPA took 60 steps and LAN more than 300). Note that LAN makes an inexplicable long detour around the point $(1.5, 0)$. Such tendencies account for the overall high cost of computations for these two algorithms as reported in Fig. 6.

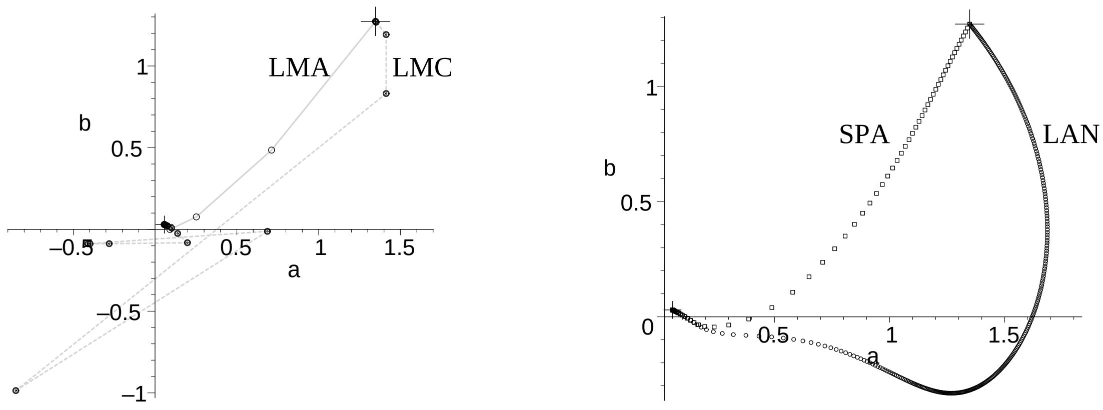


Figure 7: Paths taken by the four algorithms on the ab plane, from the initial guess at $(1.35, 1.34)$ marked by a large cross to the minimum of $\mathcal{F}$ at $(0.06, 0.03)$ (a small cross).

Next, we have repeated our experiment with a different rule of generating data samples. Instead of selecting points randomly in a square we now sample them along a circular arc of radius $R = 1$ with a Gaussian noise at level $\sigma = 0.01$. (We note that all the algorithms are invariant under translations, rotations and similarities, hence our results do not depend on the radius or the center of the circle or the location of the arc where the data are sampled.) We set the number of points to 20 and vary the arc length from 5° to 360°. Otherwise the experiment proceeded as before, including the random choice of initial guesses. The results are presented in Figures 8 and 9.

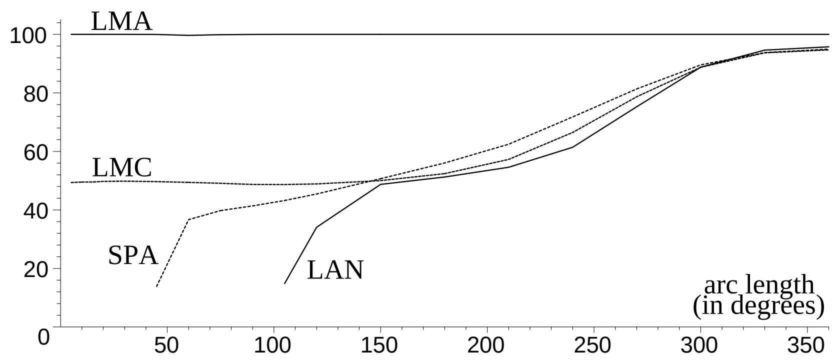


Figure 8: The probability of convergence to the minimum of $\mathcal{F}$ starting at a random initial guess.

We see that the alternative Levenberg-Marquardt method (LMA) is very robust – its displays a remarkable 100% convergence across the entire range of the arc length. The reliability of the other three methods is high (up to 95%) on full circles (360^o) but degrades to 50% on half-circles. Then the conventional Levenberg-Marquardt (LMC) stays on the 50% level for all smaller arcs down to 5^o. The Späth method breaks down on 50^o arcs and the Landau method breaks down even earlier, on 110^o arcs.

Figure 9 shows that the cost of computations for the LMA and LMC methods remains low for relatively large arcs, but it grows sharply for very small arcs (below 20^o). The LMC is generally cheaper than LMA, but, interestingly, becomes more expensive on arcs below 30^o. This shows that on small arcs the LMA beats all the other methods in both reliability and efficiency! The cost of the Späth and Landau methods is, predictably, higher than that of the Levenberg-Marquardt schemes, and it skyrockets on arcs smaller than half-circles making these two algorithms prohibitively expensive.

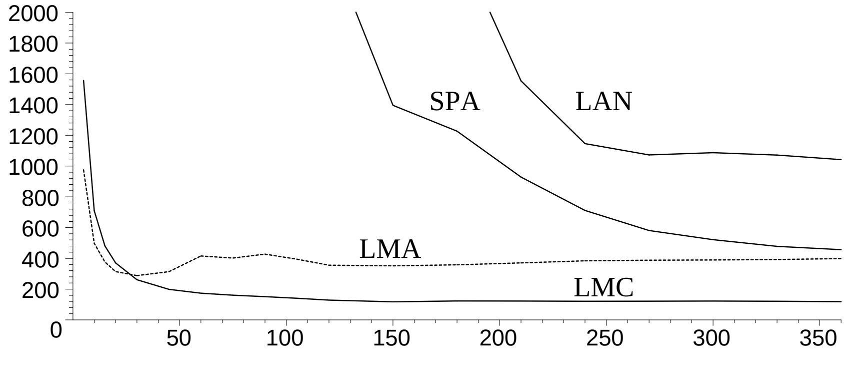


Figure 9: The average cost of computations, in flops per data point.

We emphasize, though, that our results are obtained when the initial guess is just picked randomly from a large square. In practice one can always find more sensible ways to choose an initial guess, so that the subsequent iterative schemes would perform much better than they did here. We devote the next section to various choices of the initial guess and the corresponding performance of the iterative methods.

4 Algebraic fit

The selection of an initial guess can be made by various inexpensive procedures, one of them is the so called *algebraic fit*, or AF for brevity. We will describe three different versions of the AF below.

4.1 Simple algebraic fit. The first one, we call it AF1, is a very simple and old method, it has been known since at least 1972, see [13, 19], and then rediscovered and published independently by many people [5, 12, 34, 24, 7, 36, 31]. In this method one minimizes the sum of squares of *algebraic distances*

$$\begin{aligned}\mathcal{F}_1(a,b,R) &= \sum_{i=1}^{n}[(x_i-a)^2+(y_i-b)^2-R^2]^2 \\ &= \sum_{i=1}^{n}(z_i+Bx_i+Cy_i+D)^2 \qquad (4.12)\end{aligned}$$

where $z_i = x_i^2 + y_i^2$ (as before), $B = -2a$, $C = -2b$, and $D = a^2 + b^2 - R^2$. Now, differentiating $\mathcal{F}_1$ with respect to B, C, D yields a system of linear equations, solving which gives B, C, D, and finally one computes a, b, R. This algorithm is very fast, but it is notoriously inaccurate when data are sampled along small circular arcs (it tends to grossly underestimate the radius), see [8, 27, 15].

4.2 Gradient-weighted algebraic fit (GRAF). The minimization of (4.12) is equivalent, in the alternative circle parameters A, B, C, D, to that of

$$\mathcal{F}_1(A,B,C,D) = \sum_{i=1}^{n}(Az_i+Bx_i+Cy_i+D)^2 \qquad (4.13)$$

under the constraint $A = 1$.

More generally, consider the problem of fitting a curve described by an implicit polynomial equation $P(x, y) = 0$, the coefficients of the polynomial $P(x, y)$ playing the role of parameters. The simple algebraic fit is based on minimizing

$$\mathcal{F}_{\mathrm{a}} = \sum_{i=1}^{n}[P(x_i,y_i)]^2$$

where one of the coefficients of P must be set to one to avoid the trivial solution in which all the coefficients turn zero. Our AF1 is exactly such a scheme. A better method is the so called gradient weighted algebraic fit, or GRAF for brevity. which is based on minimizing

$$\mathcal{F}_{\rm g} = \sum_{i=1}^{n} \frac{[P(x_i, y_i)]^2}{\|\nabla P(x_i, y_i)\|^2} \tag{4.14}$$

here $\nabla P(x, y)$ is the gradient of the function $P(x, y)$. There is no need to set any coefficient of P to one, since both numerator and denominator in (4.14) are homogeneous quadratic polynomials of parameters, hence the value of $\mathcal{F}_{\rm g}$ is invariant under the multiplication of all the parameters by a scalar. The reason why $\mathcal{F}_{\rm g}$ works better than $\mathcal{F}_{\rm a}$ is that we have, by the Taylor expansion,

$$\frac{|P(x_i, y_i)|}{\|\nabla P(x_i, y_i)\|} = d_i + \mathcal{O}(d_i^2)$$

where d_i is the geometric distance from the point (x_i, y_i) to the curve $P(x, y) = 0$. Thus, the function $\mathcal{F}_g$ is a linear approximation to the classical objective function $\mathcal{F}$ in (2.1).

The GRAF is known since at least 1974 [35]. It was applied specifically to quadratic curves (ellipses and hyperbolas) by Sampson in 1982 [29], and recently became standard in computer vision industry [33, 21, 11]. This method is statistically optimal, in the sense that the covariance matrix of the parameter estimates satisfies the Rao-Cramer lower bound [10, 17].

In the case of circles, $P(x, y) = A(x^2 + y^2) + Bx + Cy + D$ and $\nabla P(x, y) = (2Ax + B, 2Ay + C)$, hence

$$\begin{aligned} \|\nabla P(x_i, y_i)\|^2 &= 4Az_i^2 + 4ABx_i + 4ACy_i + B^2 + C^2 \\ &= 4A(Az_i + Bx_i + Cy_i + D) + B^2 + C^2 - 4AD \end{aligned} \tag{4.15}$$

and the GRAF reduces to the minimization of

$$\mathcal{F}_{\rm g} = \sum_{i=1}^{n} \frac{[Az_i + Bx_i + Cy_i + D]^2}{[4A(Az_i + Bx_i + Cy_i + D) + B^2 + C^2 - 4AD]^2} \tag{4.16}$$

This is a nonlinear problem that can only be solved iteratively, see some general schemes in [11, 21]. However, there are two approximations to (4.16) that lead to simpler and noniterative solutions.

4.3 Pratt's approximation to GRAF. If data points (x_i, y_i) lie close to the circle, then $Az_i + Bx_i + Cy_i + D \approx 0$, and we approximate (4.16) by

$$\mathcal{F}_2 = \sum_{i=1}^{n} \frac{[Az_i + Bx_i + Cy_i + D]^2}{B^2 + C^2 - 4AD} \tag{4.17}$$

The objective function $\mathcal{F}_2$ was proposed by Pratt [27], who clearly described its advantages over the simple algebraic fit. Pratt minimized $\mathcal{F}_2$ by using matrix methods as follows. The minimization of (4.17) is equivalent to the minimization of the simpler function (4.13) subject to the constraint $B^2 + C^2 - 4AD = 1$. We write the function (4.13) in matrix form as $\mathcal{F}_1 = \mathbf{A}^T\mathbf{M}\mathbf{A}$, where $\mathbf{A} = (A, B, C, D)^T$ is the vector of parameters and $\mathbf{M}$ is the matrix of moments:

$$\mathbf{M} = \begin{pmatrix} M_{zz} & M_{xz} & M_{yz} & M_z \\ M_{xz} & M_{xx} & M_{xy} & M_x \\ M_{yz} & M_{xy} & M_{yy} & M_y \\ M_z & M_x & M_y & n \end{pmatrix} \tag{4.18}$$

where M_{xx}, M_{xy}, etc. denote moments, for example $M_{xx} = \sum x_i^2$, $M_{xy} = \sum x_i y_i$. Note that $\mathbf{M}$ is symmetric and positive semidefinite (actually, $\mathbf{M}$ is positive definite unless the data points are interpolated by a circle or a line). The constraint $B^2 + C^2 - 4AD = 1$ can be written as $\mathbf{A}^T\mathbf{B}\mathbf{A} = 1$, where

$$\mathbf{B} = \begin{pmatrix} 0 & 0 & 0 & -2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -2 & 0 & 0 & 0 \end{pmatrix} \tag{4.19}$$

Now introducing a Lagrange multiplier η we minimize the function $\mathcal{F}_* = \mathbf{A}^T\mathbf{M}\mathbf{A} - \eta(\mathbf{A}^T\mathbf{B}\mathbf{A} - 1)$. Differentiating with respect to $\mathbf{A}$ gives $\mathbf{M}\mathbf{A} = \eta\mathbf{B}\mathbf{A}$. Hence η is a generalized eigenvalue for the matrix pair (*pencil of matrices*) $(\mathbf{M}, \mathbf{B})$ and $\mathbf{A}$ the corresponding generalized eigenvector. The matrix $\mathbf{B}$ is symmetric and has four real eigenvalues $\{1, 1, 2, -2\}$. If $\mathbf{M}$ is positive definite, by Sylvester's law of inertia the generalized eigenvalues of the pencil $(\mathbf{M}, \mathbf{B})$ are all real, exactly three of them are positive and one is negative. To determine which one corresponds to the minimum of $\mathcal{F}_2$ we observe that $\mathcal{F}_2 = \mathbf{A}^T\mathbf{M}\mathbf{A} = \eta\mathbf{A}^T\mathbf{B}\mathbf{A} = \eta$, hence the minimum of $\mathcal{F}_2$ corresponds to the *smallest nonnegative* generalized eigenvalue.

Generalized eigenpairs $(\eta, \mathbf{A})$ can be found by standard matrix methods. Those, however, tend to be computationally extensive [27] in experiments involving mass data processing (for example, in high energy physics one often needs to process millions of track images within days or hours). In this case a faster alternative to standard matrix methods is solving the quartic equation $Q_4(\eta) := \det(\mathbf{M} - \eta\mathbf{B}) = 0$ by Newton's method starting at $\eta = 0$. Newton's iterations are guaranteed to converge to the smallest nonnegative root $\eta_* = \min\{\eta \geq 0 : \ Q_4(\eta) = 0\}$, because the polynomial $Q_4(\eta)$ is decreasing and concave up between 0 and η_*. A detailed analysis of the polynomial $Q_4(\eta)$ is provided on our web page [9].

We denote the above algorithm by AF2.

4.4 Taubin's approximation to GRAF. Another way to simplify (4.16) is to average the variables in the denominator:

$$\mathcal{F}_3 = \sum_{i=1}^{n} \frac{[Az_i + Bx_i + Cy_i + D]^2}{[4A^2\langle z\rangle + 4AB\langle x\rangle + 4AC\langle y\rangle + B^2 + C^2]^2} \tag{4.20}$$

where $\langle z\rangle = (\sum_{i=1}^n z_i)/n = M_z/n$, $\langle x\rangle = M_x/n$, and $\langle y\rangle = M_y/n$. This idea was first proposed by Agin [1] but became popular after a publication by Taubin [33], and it is known now as Taubin's method.

The minimization of (4.20) is equivalent to the minimization of $\mathcal{F}_1$ defined by (4.13) subject to the constraint

$$4A^2M_z + 4ABM_x + 4ACM_y + B^2n + C^2n = 1 \tag{4.21}$$

This problem can be expressed in matrix form as $\mathcal{F}_1 = \mathbf{A}^T\mathbf{M}\mathbf{A}$, see notation in 4.3, with the constraint equation $\mathbf{A}^T\mathbf{C}\mathbf{A} = 1$, where

$$\mathbf{C} = \begin{pmatrix} 4M_z & 2M_x & 2M_y & 0 \\ 2M_x & n & 0 & 0 \\ 2M_y & 0 & n & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{4.22}$$

is a symmetric and positive semidefinite matrix. Introducing a Lagrange multiplier η, as in 4.3, we find that $(\eta, \mathbf{A})$ is a generalized eigenpair for the matrix pencil $(\mathbf{M}, \mathbf{C})$, and the minimum of $\mathcal{F}_3$ corresponds to the smallest nonnegative generalized eigenvalue η.

Generalized eigenpairs $(\eta, \mathbf{A})$ can be found by standard matrix methods. Alternatively, as in 4.3, one can solve the cubic equation $\det(\mathbf{M} - \eta\mathbf{C}) = 0$ by Newton's method starting at $\eta = 0$. Newton's iterations are guaranteed to converge to the smallest nonnegative generalized eigenvalue.

We denote the above algorithm by AF3.

4.5 Nonalgebraic (heuristic) initializations. Some experimenters also use various simplistic procedures to initialize an iterative scheme. For example, some pick three data points that are sufficiently far apart and find the interpolating circle [16]. Others place the initial center of the circle at the centroid of the data [2]. Even though such tricks are generally inferior to the algebraic fits, we will include two of them in our experimental tests for comparison. We call them TRI and CEN:

- TRI: Find three data points that make the triangle of maximum area and construct the interpolating circle.

- CEN: Put the center of the circle at the centroid of the data and then compute the radius by (2.4).

4.5 Experimental tests. Here we combine the fitting algorithms in pairs: first, an algebraic (or heuristic) prefit yields a circle, then an iterative algorithm uses it as the initial guess and proceeds to minimize the objective function $\mathcal{F}$. Our goal here is to evaluate the performance of the iterative methods described in Section 3 when they are initialized by various algebraic (or heuristic) prefits, rather than completely randomly. We test all 5 initialization methods – AF1, AF2, AF3, TRI, and CEN – combined with all 4 iterative schemes – LMA, LMC, SPA, and LAN (in the notation of Section 3) – a total of $5 \times 4 = 20$ pairs.

We conduct two major experiments, as we did in Sect. 3.3. First, we generate 10000 samples of n points randomly with a uniform distribution in the unit square $0 < x, y < 1$. For each sample, we determine the global minimum of the objective function $\mathcal{F}$ by running the most reliable iterative scheme, LMA, starting at 1000 random initial guesses, as we did in Section 3.3. This is a credible (though, expensive) way to locate the global minimum of $\mathcal{F}$. Then we apply all 20 pairs of algorithms to each sample. Note that no pair needs an initial guess, since the first algorithm in each pair is just designed to provide one.

After running all $N = 10000$ samples, we find, for each pair $[ij]$ of algorithms ($1 \leq i \leq 5$ and $1 \leq j \leq 4$), the number of samples, N_{ij}, on which that pair successfully converged to the global minimum of $\mathcal{F}$. The ratio N_{ij}/N then represents the probability of convergence to the minimum of $\mathcal{F}$ for the pair $[ij]$. We also find the average number of iterations the convergence takes, for each pair $[ij]$ separately.

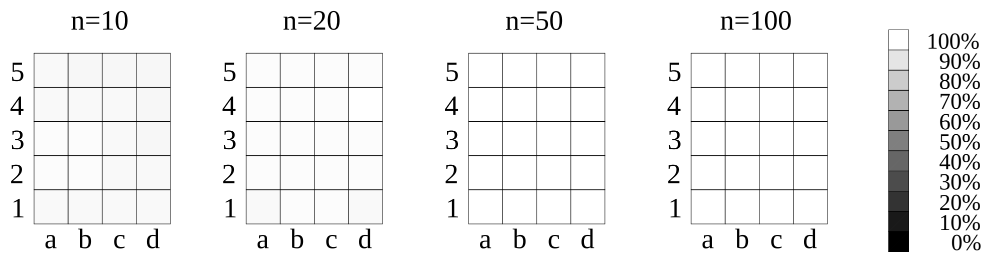


Figure 10: The probability of convergence to the minimum of $\mathcal{F}$ for 20 pairs of algorithms. The bar on the right explains color codes.

This experiment was repeated for $n = 10, 20, 50$, and 100 data points. The results are presented in Figures 10 and 11 by grey-scale diagrams. The bars on the right explain our color codes. For brevity, we numbered the algebraic/heuristic methods:

$$1 = \text{AF1}, \;\; 2 = \text{AF2}, \;\; 3 = \text{AF3}, \;\; 4 = \text{TRI}, \;\; \text{and } 5 = \text{CEN}$$

and labelled the iterative schemes by letters:

$$\text{a} = \text{LMA}, \;\; \text{b} = \text{LMC}, \;\; \text{c} = \text{SPA}, \;\; \text{and d} = \text{LAN}$$

Each small square (cell) represents a pair of algorithms, and its color corresponds to its characteristic according to our code.

Fig. 10 shows the probability of convergence to the global minimum of $\mathcal{F}$. We see that all the cells are white or light grey, meaning the reliability remains close to 100% (in fact, it is 97-98% for $n = 10$, 98-99% for $n = 20$ and almost 100% for $n \geq 50$.)

We conclude that for completely random samples filling the unit square uniformly, all five prefits provide a good initial guess, so that any subsequent iterative method has little trouble converging to the minimum of $\mathcal{F}$.

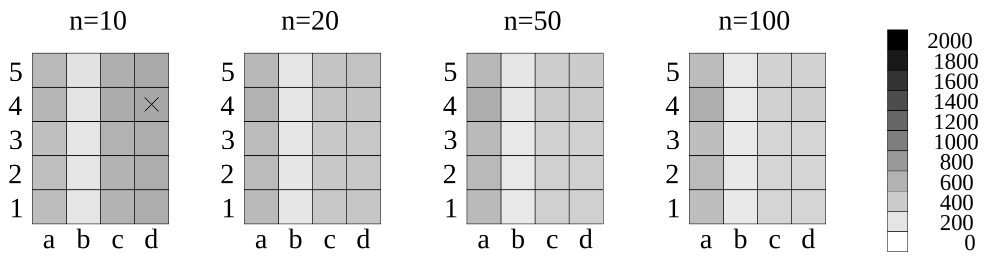


Figure 11: The cost in flops per data point. The bar on the right explains color codes.

Fig. 11 shows the computational cost for all pairs of algorithms. Colors represent the number of flops per data point, as coded by the bar on the far right. We see that the cost remains relatively low, in fact it never exceeds 700 flops per point (compare this to Figs. 6 and 9). The highest cost here is 684 flops per point for the pair TRI+LAN and $n = 10$ points, marked by a cross.

The overall good performance observed in the above experiment is due to the fact that random samples make, in a sense, an "easy prey" to fitting algorithms. Indeed, when the data points are scattered chaotically, the objective function $\mathcal{F}$ has no pronounced minima or valleys, its shape is usually simple. Even an inaccurate initial guess allows the iterative schemes to converge to the minimum rapidly.

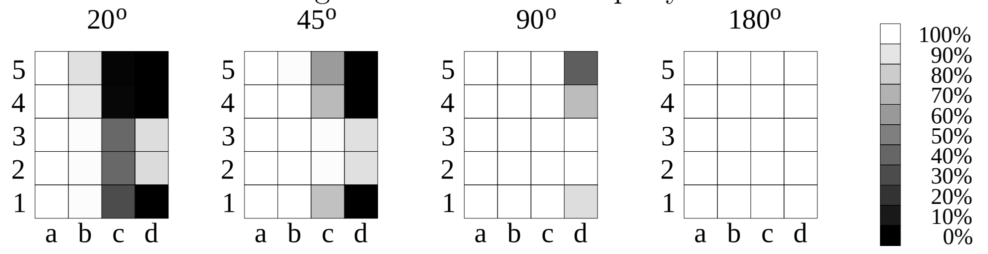


Figure 12: The probability of convergence to the minimum of $\mathcal{F}$ for 20 pairs of algorithms. The bar on the right explains color codes.

We now turn to the second experiment, where data points are sampled, as in Sect. 3.3, along a circular arc of radius $R = 1$ with a Gaussian noise at level $\sigma = 0.01$. We set the number of points to $n = 20$ and vary the arc length from 20° to 180°. In this case the objective function usually has a sharp narrow minimum or a deep narrow valley, and so the iterative schemes heavily depend on an accurate initial guess.

The results of this second experiment are presented in Figures 12 and 13, in the same fashion as those in Figs. 10 and 11. We see that the performance is not as good as before and deteriorates as the arc length decreases. The probability of convergence to the minimum of $\mathcal{F}$ sometimes drops to zero (black cells on Fig. 12), and the cost per data point exceeds our maximum of 2000 flops (black cells on Fig. 13).

First, let us compare the iterative schemes. The Späth and Landau methods (columns c and d) become unreliable and too expensive on small arcs. Interestingly, though, Landau somewhat outperforms Späth here, while in earlier experiments reported in Sect. 3.3, Späth fared better. Both Levenberg-Marquartd schemes (columns a and b) are quite reliable and fast across the entire range of the arc length from 180^o to 20^o.

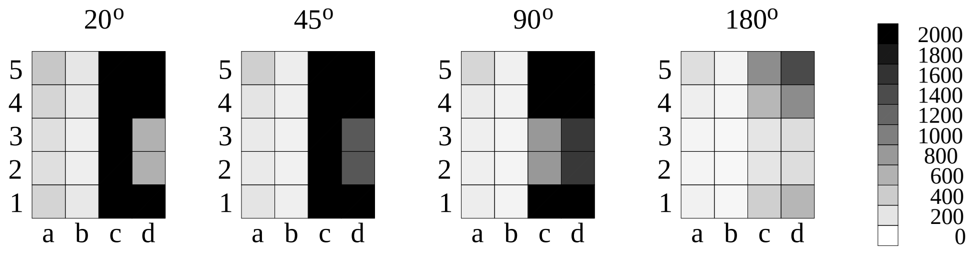


Figure 13: The cost in flops per data point. The bar on the right explains color codes.

This experiment clearly demonstrates the superiority of the Levenberg-Marquardt method over fixed-point iterative schemes such as Späth or Landau. The latter two, even if supplied with the best possible prefits, tend to fail or become prohibitively expensive on small circular arcs.

Lastly, we extend this test to even smaller circular arcs (between 5 and 15 degrees) keeping only the two Levenberg-Marquardt schemes in our race. The results are presented in Figure 14. We see that, as the arc gets smaller, the conventional Levenberg-Marquardt (LMC) gradually loses its reliability but remains efficient, while the alternative scheme (LMA) gradually slows down but remains extremely reliable. Interestingly, both schemes take about the same number of iterations to converge, for example, on 5^o arcs the pair AF2+LMA converged in 19 iterations, on average, while the pair AF2+LMC converged in 20 iterations. The higher cost of the LMA seen on Fig. 14 is entirely due to its complexity per iteration. Still, the cost of LMA per data point remains moderate, it is nowhere close to our maximum of 2000 flops (in fact, it always stays below 1000 flops).

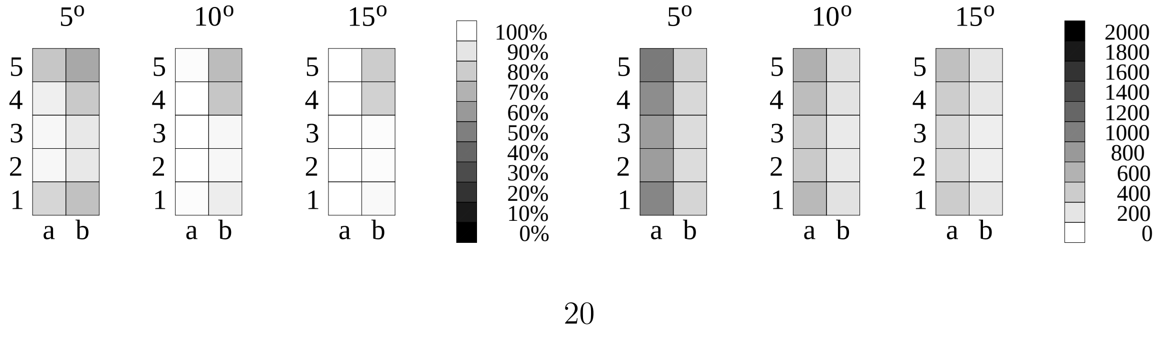


Figure 14: The probability of convergence (left) and the cost in flops per data point (right) for the two Levenberg-Marquardt schemes.

Our experiments were done on Pentium IV personal computers and a Dell Power Edge workstation with 32 nodes of dual 733MHz processors at the University of Alabama at Birmingham.

References

[1] G.J. Agin, Fitting Ellipses and General Second-Order Curves, Carnegi Mellon University, Robotics Institute, Technical Report 81-5, 1981.

[2] S.J. Ahn, W. Rauh, and H.J. Warnecke, Least-squares orthogonal distances fitting of circle, sphere, ellipse, hyperbola, and parabola, *Pattern Recog.* **34**, 2001, 2283–2303.

[3] M. Berman and D. Culpin, The statistical behaviour of some least squares estimators of the centre and radius of a circle, *J. R. Statist. Soc. B*, **48**, 1986, 183–196.

[4] R.H. Biggerstaff, Three variations in dental arch form estimated by a quadratic equation, *J. Dental Res.* **51**, 1972, 1509.

[5] F.L. Bookstein, Fitting conic sections to scattered data, *Comp. Graph. Image Proc.* **9**, 1979, 56–71.

[6] Y.T. Chan and S.M. Thomas, Cramer-Rao Lower Bounds for Estimation of a Circular Arc Center and Its Radius, *Graph. Models Image Proc.* **57**, 1995, 527–532.

[7] B.B. Chaudhuri and P. Kundu, Optimum Circular Fit to Weighted Data in Multi-Dimensional Space, *Patt. Recog. Lett.* **14**, 1993, 1–6.

[8] N.I. Chernov and G.A. Ososkov, Effective algorithms for circle fitting, *Comp. Phys. Comm.* **33**, 1984, 329–333.

[9] N. Chernov and C. Lesort, Fitting circles and lines by least squares: theory and experiment, preprint, available at http://www.math.uab.edu/cl/cl1

[10] N. Chernov and C. Lesort, Statistical efficiency of curve fitting algorithms, preprint, available at http://www.math.uab.edu/cl/cl2

[11] W. Chojnacki, M.J. Brooks, and A. van den Hengel, Rationalising the renormalisation method of Kanatani, *J. Math. Imaging & Vision* **14**, 2001, 21–38.

[12] J.F. Crawford, A non-iterative method for fitting circular arcs to measured points, *Nucl. Instr. Meth.* **211**, 1983, 223–225.

[13] P. Delonge, Computer optimization of Deschamps' method and error cancellation in reflectometry, in *Proceedings IMEKO-Symp. Microwave Measurement (Budapest), 1972*, 117–123.

[14] P. R. Freeman, Note: Thom's survey of the Avebury ring, *J. Hist. Astronom.* **8**, 1977, 134–136.

[15] W. Gander, G.H. Golub, and R. Strebel, Least squares fitting of circles and ellipses, *BIT* **34**, 1994, 558–578.

[16] S. H. Joseph, Unbiased Least-Squares Fitting Of Circular Arcs, *Graph. Models Image Proc.* **56**, 1994, 424–432.

[17] K. Kanatani, Cramer-Rao lower bounds for curve fitting, *Graph. Models Image Proc.* **60**, 1998, 93–99.

[18] V. Karimäki, Effective circle fitting for particle trajectories, *Nucl. Instr. Meth. Phys. Res. A* **305**, 1991, 187–191.

[19] I. Kasa, A curve fitting procedure and its error analysis, *IEEE Trans. Inst. Meas.* **25**, 1976, 8–14.

[20] U.M. Landau, Estimation of a circular arc center and its radius, *Computer Vision, Graphics and Image Processing* **38**, 1987, 317–326.

[21] Y. Leedan and P. Meer, Heteroscedastic regression in computer vision: Problems with bilinear constraint, *Intern. J. Comp. Vision* **37**, 2000, 127–150.

[22] K. Levenberg, A Method for the Solution of Certain Non-linear Problems in Least Squares, *Quart. Appl. Math.* **2**, 1944, 164–168.

[23] D. Marquardt, An Algorithm for Least Squares Estimation of Nonlinear Parameters, *SIAM J. Appl. Math.* **11**, 1963, 431–441.

[24] L. Moura and R.I. Kitney, A direct method for least-squares circle fitting, *Comp. Phys. Comm.* **64**, 1991, 57–63.

[25] G. A. Ososkov, JINR technical report P10-83-187, Dubna, 1983, pp. 40 (in Russian).

[26] K. Paton, Conic sections in chromosome analysis, *Pattern Recogn.* **2**, 1970, 39–51.

[27] V. Pratt, Direct least-squares fitting of algebraic surfaces, *Computer Graphics* **21**, 1987, 145–152.

[28] S.M. Robinson, Fitting spheres by the method of least squares, *Commun. Assoc. Comput. Mach.* **4**, 1961, 491.

[29] P.D. Sampson, Fitting conic sections to very scattered data: an iterative refinement of the Bookstein algorithm, *Comp. Graphics Image Proc.* **18**, 1982, 97–108.

[30] C. Shakarji, Least-squares fitting algorithms of the NIST algorithm testing system, *J. Res. Nat. Inst. Stand. Techn.* **103**, 1998, 633–641.

[31] H. Spath, Least-Squares Fitting By Circles, *Computing* **57**, 1996, 179–185.

[32] H. Spath, Orthogonal least squares fitting by conic sections, in *Recent Advances in Total Least Squares techniques and Errors-in-Variables Modeling*, SIAM, 1997, pp. 259–264.

[33] G. Taubin, Estimation Of Planar Curves, Surfaces And Nonplanar Space Curves Defined By Implicit Equations, With Applications To Edge And Range Image Segmentation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **13**, 1991, 1115–1138.

[34] S.M. Thomas and Y.T. Chan, A simple approach for the estimation of circular arc center and its radius, *Computer Vision, Graphics and Image Processing* **45**, 1989, 362–370.

[35] K. Turner, *Computer perception of curved objects using a television camera*, Ph.D. Thesis, Dept. of Machine Intelligence, University of Edinburgh, 1974.

[36] Z. Wu, L. Wu, and A. Wu, The Robust Algorithms for Finding the Center of an Arc, *Comp. Vision Image Under.* **62**, 1995, 269–278.

[37] Z. Zhang, Parameter Estimation Techniques: A Tutorial with Application to Conic Fitting, *International Journal of Image and Vision Computing*, **15**, 1997, 59–76.