IOSR Journal of Mathematics (IOSR-JM)
e-ISSN: 2278-5728, p-ISSN: 2319-765X. Volume 14, Issue 3 Ver.I (May - June 2018), PP 12-18
www.iosrjournals.org

The Geometric Least Squares Fitting Of Ellipses

Abdellatif Bettayeb
Department of General Studies, Jubail Industrial College, Jubail Industrial City, Kingdom of Saudi Arabia
Corresponding Author: Abdellatif Bettayeb

Abstract: *The problem of Fitting conic sections to given data in the plane is one which is of great interest and arises in many applications, e.g. computer graphics, statistics, coordinate metrology, aircraft industry, metrology, astronomy, refractometry, and petroleum engineering [7, 2, 3]. In this paper, we present several methods which have been suggested for Fitting ellipses to data in the plane. We will look particularly at one method, by giving examples and using Matlab to solve these problem.*

Date of Submission: 16-04-2018 Date of acceptance: 04-05-2018

I. Introduction

Let a relationship between variables *x* and *y* be given by $f(x, y; p) = 0,$ where $p \in R^n$ is a vector of parameters. For example, this could be an ellipse or any conic in the *x,y* plane.
Let data points $(x_i, y_i),\ i = 1,..., m$ be given. Then ideally we wish to choose p so that
$f(x_i, y_i; p) = 0, \quad i = 1,....m.$

However, this is unlikely to be possible, so we need some other ways of choosing p.
This method is known as geometric fitting and uses the parametric representation of the ellipse such that the sum of the squared orthogonal distances from each data points to the ellipse is minimal, and this is discussed by Helmut Spath in [4].

In the other sections we introduces different numerical different numerical examples, with relevant figures and results.

II. Geometric Fitting

Given the model
$f(x, y; \beta) = 0,$ (2.1)
and the data points $(x_i, y_i),\ i = 1,..., m$ in the plane, another possibility is as follows:

We can choose the sum of the squares of the distances from the data points (x_i, y_i) to the curve $f(x, y; \beta) = 0$ to be minimized. We consider the special case when it is possible to give a parameterization of the curve, using $x = x(t),\ y = y(t)$. Then we examine here a special algorithm proposed by Helmut Spath. A more general Gauss-Newton method is considered also to be compared with it. Both methods are applied for ellipses.

III. The method of Spat for an ellipse

Let the data points $(x_k, y_k),\ k = 1,..., m$ be given in the plane, and $x = x(t),\ y = y(t)$ be the parametric representation of an ellipse
$x(t) = a + p\cos t, \quad y(t) = b + q\sin t.$
We have to minimize with respect to β and t

$$S(\beta, t) = \sum_{i=1}^{m} [(x_i - x(t_i))^2 + (y - y(t_i))^2], \tag{3.1}$$

where $\beta = [a\ b\ p\ q]^T$ is the parameter vector which must be determined. Also $t = (t_1,..., t_m)^T$ has to be determined, representing the positions of the points on the curve whose distances to the data points are minimized i.e. the distances are orthogonal.
Notice that t_i only appears in the i-th term i.e. $S(\beta, t)$ is separable with

respect to the unknowns $t_1,..., t_m$.
Conditions for a minimum are

$$\frac{\partial S}{\partial t_i}(\beta,t)=0, \quad (i=1,\dots,m). \qquad (3.2)$$

$$\frac{\partial S}{\partial \beta_j}(\beta,t)=0, \quad (j=1,\dots,n). \qquad (3.3)$$

If β is fixed, the (3.2) corresponds to m equations, each of which can be solved for t_i. If t is fixed, the (3.3) is a linear least squares problem for β. We can obtain a solution by an alternation procedure, updating t and β systematically.
First we fix β and we must determine the minimum t_i which satisfies (3.2). That is

$$2(x_i - x(t_i))(-x'(t_i)) + 2(y_i - y(t_i))(-y'(t_i)) = 0, \; (j=1,..., m). \quad (3.4)$$

For each i, then we could select the t_i that globally minimises the i-th term of S.

Thus, for given β, we could attain a global minimum of $S(\beta,t)=S_1$ with respect to *t*.

Now for the ellipse it turns out that the equations (3.4) already are or can be transformed into m polynomial equations of low degree less than or equal to four. This is explained below. For the ellipse we have four roots (zeros), but either two or four are real zeros.
Next we fix t at these values and satisfy (3.3), which can be interpreted as a linear squares problem, because appears linearly.

This delivers the global minimum for $S(\beta, t)$ as desired, for this *t*.
Each step gives a reduction of the value of *S* until no further reduction is possible, when a minimum of (3.1) has seen found.
Consider the calculation for β. Let *r* be the residual vector:

$$r = [r_x \;\; r_y]^T,$$

each component of *r* is:

$$r_x = \begin{bmatrix} x_1 - a - p\,cost_1 \\ \vdots \\ x_m - a - p\,cost_m \end{bmatrix},$$

$$r_y = \begin{bmatrix} y_1 - b - q\,cost_1 \\ \vdots \\ y_m - b - q\,cost_m \end{bmatrix},$$

we can write the i-th component of *r* as follows

$$\sum_{k=1}^{n} c_{ik}\beta_k + d_i = r_i \qquad (i=1,\dots,m) \quad m>n,$$

if we introduce the residuals r_i, in matrix form, we can write the error equation

$$C\beta + d = r, C \in R^{2m\times n}, \quad \beta \in R^n, \quad d, r \in R^{2m}, \qquad (3.5)$$

where *C* is a *2m x n*, matrix, with *n* in this case equals to 4, and d is a vector of length *2m*.

$$C = \begin{bmatrix} 1 & 0 & \cos t_1 & 0 \\ \vdots & & & \vdots \\ 1 & 0 & \cos t_m & 0 \\ 0 & 1 & 0 & \sin t_1 \\ \vdots & & & \vdots \\ 0 & 1 & 0 & \sin t_m \end{bmatrix}, d = \begin{bmatrix} x_1 \\ \vdots \\ x_m \\ y_1 \\ \vdots \\ y_m \end{bmatrix},$$

we assumed that the matrix C has the maximal rank, i.e. its column vectors are linearly independent.

The unknowns β_k of the error equations are to be determined according to the Gaussian principle, such that *S* the sum of squares of the residuals *r* is minimal. And this is equivalent to minimising the square of the Euclidean
norm of the residual vector.

From (3.5) we obtain

$$r^T r = (C\beta + d)^T (C\beta + d) = \beta^T C^T C\beta + \beta^T C^T d + d^T C\beta + d^T d$$
$$= \beta^T C^T C\beta + 2(C^T d)^T \beta + d^T d.$$

We put $A = C^T C \quad b = C^T d \quad A \in R^{n\times n}, \quad b \in R^n.$

As *C* has maximal rank, the symmetric matrix *A* is positive definite. Thus

$$S_2 = S(\beta, t) := r^T r = \beta^T A\beta + 2b^T \beta + d^T d = Min! \qquad (3.6).$$

A necessary condition for minimising *S*(β) at the point , is the gradient $\nabla S(\beta) = 0.$

The i-th component of the gradient $\nabla S(\beta)$ is obtained from the explicit representation of (3.6).

$$\frac{\partial S}{\partial \beta_i}(\beta, t) = 2\sum_{k=1}^{n} a_{ik}\beta_k + 2b_i \qquad (i = 1,2,..., n) \qquad (3.7)$$

After division by 2 from (3.7), we obtain the linear system of equations:

$$A\beta + b = 0, \qquad (3.8)$$

for the unknowns $\beta_1, \beta_2,..., \beta_n.$ We call this system of equations the *Normal equations of the error equations* (3.8). The Matrix A is positive definite, thus from the assumption on C, the unknowns $_k$ are uniquely determined by the normal equations (3.8).

The function S(β) is indeed minimised by these values, because the Hessian matrix of S( ), the matrix of the second partial derivatives, is equal to the positive definite matrix A (see [6] page 294-296).

There are a lot of ways to solve the normal equations (3.8). Because the direct method used by MATLAB is faster, we solve the normal equation by this method

$$\beta = A \setminus (-b) = 0.$$

There is also the possibility that (3.5) can be solved directly using the (\)
MATLAB command.

3.1 The general Algorithm

Minimise S(β,t) w.r.t t, let S_1 be the result for this stage.

Step 1:

Give initial guess $_0$, tolerancetol.

Step 2:

Compute by (Maple) the derivative of *S* w.r.t. (t)

1. $S_i := (x_i - a - p * \cos(t_i))^2 + (y_i - b - q * \sin(t_i))^2 = 0;$
2. $eq := (x_i - a - p * \cos(t_i))^2 + (y_i - b - q * \sin(t_i))^2 = 0;$
3. $sol := solve(eq, t);$

$$sol := 2 * \arctan(RootOf((-q * b + q * y_i) * Z^4 + (2 * p * x_i - 2 * p * a + 2 * p^2 - 2 * q^2) * Z^3$$
$$+ (2 * p * x_i - 2 * p * a - 2 * p^2 + 2 * q^2) * z + q * b - q * y_i))$$

$$B=[-q*b+q*y(i);2*p*x(i)-2*p*a+2*p^2-2*q^2;0;2*p*x(i)-2*p*a-2*p^2+$$
$$+2*q^2;q*b-q*y(i)];\quad for\ i=1:n.$$

$$Alpha=roots(B);$$

$$T=2a\tan(Alpha);\quad (in\ MATLAB).$$

4. Take only the real values of T
5. Substitute this real values in S_i, and choose the minimum of S_i and the correspondent T_i.
6. The minimal S is $\sum_{i=1}^{n} S_i$, equals to S_1.

Step 3:
Fix t and find S_2 the minimium of S w.r.t. β
1. Solve the linear least square problem A*β = b as we saw above such that = A\ - b.
2. $S_2 = r' * r,$ where *r* is the residual defined before.
3. Compute S_1 and S_2 until we get the minimum value of them with tolerance proposed. At each iteration, there must be a decrease in the value of S.

Remark: Direct methods are usually faster and more generally applicable, the usual way to access direct methods in MATLAB is not through the LU or Cholesky factorisation, but rather with the matrix division operator / and \. If A is square, the result of X = A\B is the solution to the linear system AX = B. If A is not square, then a least squares solution is computed (see [5] and MATLAB version 4.2c, 1994).

IV. Gauss-Newton method

On considering the nonlinear least squares problem:

$$\min_{X\in R^{n+m}} S=\sum_{i=1}^{m} r_i^2(X).$$

For the ellipse $X=[\beta,t]^T$, where $\beta=[a\ b\ p\ q]^T$ and $t=[t_1\cdots t_m]^T$, , n=4,
$r=[r_1\ r_2\ldots r_m]^T$, $S=r^T r.$

Differentiating w.r.t. X_i, $(i=1,..., n+m)$

$$\frac{\partial S}{\partial X_i}=2\sum_{j=1}^{m} r_j \frac{\partial r_j}{\partial X_i}. \qquad (4.1)$$

$$\nabla S(X)=2J^T r,$$

where J is the Jacobian matrix associated with S and is an m x n matrix of the form:

$$J_{pq}\equiv\frac{\partial r_p}{\partial X_q}.$$

Hence the p-th row is the derivative vector of the p-th sub-function r_X w.r.t. each element of X.

Differentiate again:

$$\frac{\partial^2 S}{\partial X_i \partial X_j}=2\{\sum_{j=1}^{m}[\frac{\partial r_j}{\partial X_k}.\frac{\partial r_j}{\partial X_i}+r_j\frac{\partial^2 r_j}{\partial X_i \partial X_k}]\}, \qquad (4.2)$$
$$\nabla^2 S(X)=2(J^T J+B),$$

Where $\nabla^2 S$ is the *n x n* symmetric Hessian matrix of *S*. The *n x n* matrix *B* which the error matrix is:

$$B = \sum_{i=1}^{m} r_i \nabla^2 r_i ,$$

where $\nabla^2 S$ symmetric and positive definite $J^T J$ positive semi-definite because $z^T J^T J z \equiv y^T y \geq 0,$ thus we neglect B [1].

4.1 **The general algorithm for Gauss-Newton Method**

Then the G-N Algorithm is:

Step 1:
Choose x^0 initial approximation to x and a maximum value of *S* let be $S_1 = 10^8$, and a tolerance *tol*, set $k = 0$.

Step 2:
Compute r^k, J^k, thus $J^{kT}J^k$ and $J^{kT} r^k$.
If $\|J^T r\| \leq tol,$ stop.
$S = r^{kT} * r^k;$
$d = abs(S1 - S);$
If $d < tol$, break, end

Step 3:
Solve the equations by finding δ^k, here δ is the correction vector
$J^{kT} J^k \delta^k = -J^{kT} r^k .$

Step 4:
If $\|\delta^k\| < tol,$ return. Otherwise set $x^{k+1} = x^k + \alpha^k \delta^k, k = k+1.$ We put here the step length or the damping factor equals to 1 (for the ellipse and the circle).

Step 5:
If $S^{k+1} \leq S^k$ return to step 2. where $S = r^T r$. Otherwise set $\alpha = \frac{\alpha}{2}$. i.e. halve the step-length , until we get $S^{k+1} \leq S^k$ then return to step 4, (this case appears clearly for the parabola).
The correction vector δ^k is based on local information, the new approximation may have undesirable properties. For example, even though
δ^k is not uphill at x^k, we may still find that $S^{k+1} > S^k$ (the case of the parabola). It is necessary, therefore, to introduce a factor α^k, which modifies the norm of the correction vector; it becomes convenient to refer to the latter as a " search direction ". And α^k is usually called a step length, or in the present context, a " damping factor " (see [1]).

V. Examples

With different set of points, we applied here the geometric methods (Spath, Gauss-Newton) for an ellipse. MATLAB was used here, because it is easy to implement against another package or language like Fortran, and we save a lot of time too.

5.1Fitting Ellipses
Consider the 'Spath' data set given from [4] in Table (4.1) which is used for all examples of ellipses.

x	8	3	2	7	6	6	4
y	1	6	3	7	1	10	0

5.2 Example 1: Geometric method with Spath Algorithm

Figure 1 shows the ellipse generated from the data using the Spath method with initial guess $\beta_0 = [5.2604 \quad 4.4887 \quad 2.9542 \quad 4.8224]'$. .

And s= 2.4279 number of iterations = 22.

The ellipse generated is

$$x = 5.041206 + 2.626025\cos\theta$$

$$y = 4.790501 + 5.325467\sin\theta$$

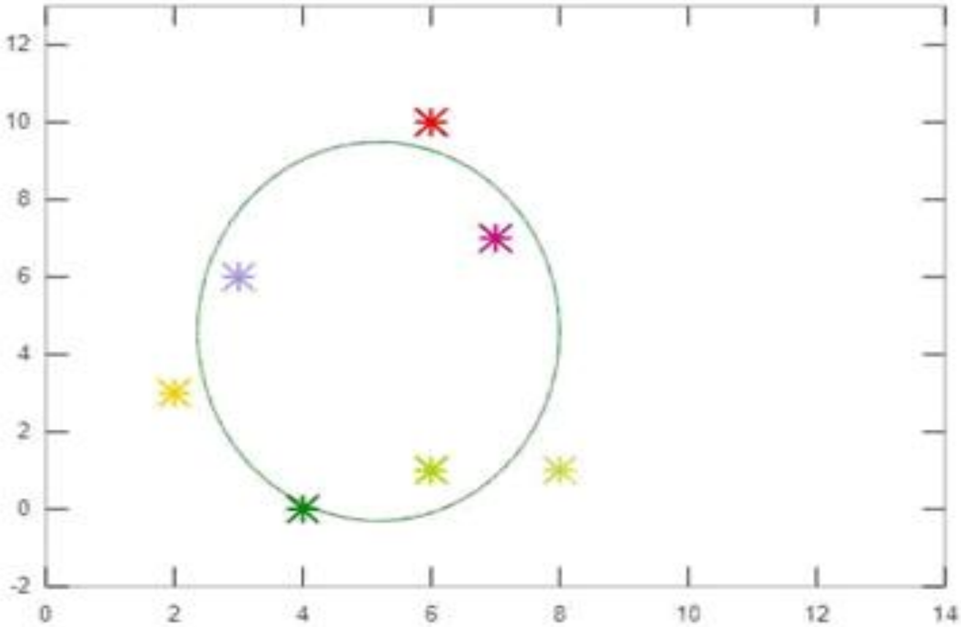

Figure 1: Ellipse fits with Spath Method

5.3 Example 2: Geometric method with Gauss-Newton

We get the same Figure as Figure 1 and we got the results as follow.

s= 2.4279 number of iterations = 10.

The ellipse generated is:

$$x = 5.041063 + 2.625540\cos\theta$$

$$y = 4.789972 + 5.326559\sin\theta$$

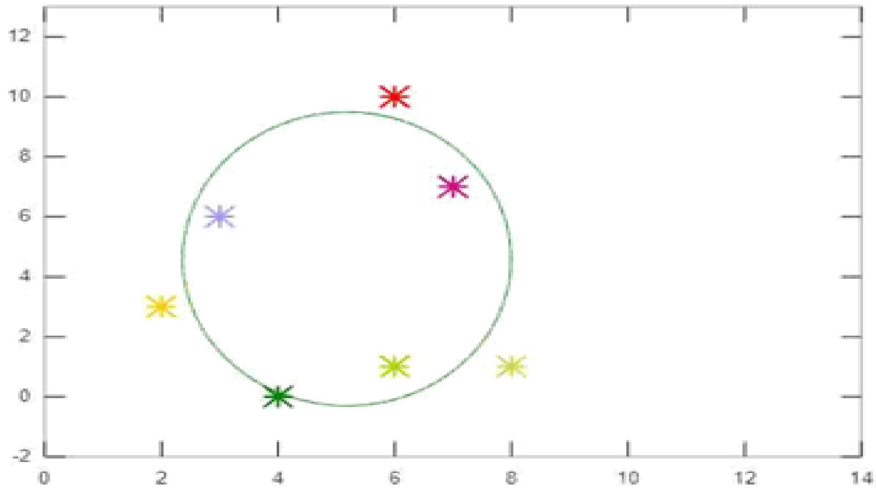

Figure 2: Ellipse fits with Gauss-Newton Method.

Bibliography

[1] Dixon, L. C. W., Spedicato, E. and Szego, G. P., Nonlinear Optimization: theory and algorithms, Birkhauser Boston, 1980. ISBN 3-7643-3020-1. pp. 91-102.
[2] Fletcher, R. G., Practical Methods of Optimization, John Wiley and Sons, 1995. ISBN 0 471 91547 5 pp. 111-136.
[3] Gander, W., Golub, G. H. and Strebel, R., Fitting of circles and ellipses: least square solution,BIT, 34(1994), pp. 556-577.
[4] Huffel Sabine Van, Recent Advances in total least squares techniques and errors in variables modeling, " Orthogonal Least Squares Fitting by Conic Sections " by Helmuth Spath pp. 259-264, Library of Congress, USA, 1997. ISBN 0-89871-393-5.
[5] Kolman, B., Elementary linear algebra, 1991. ISBN 0-02-366045-7.
[6] Schwarz, H. R., Numerical Analysis: A comprehensive introduction, Great Britain, 1989. ISBN 0471 92064 9 pp. 294-329.
[7] The MathWorks, Inc. MATLAB REFERENCE GUIDE, 1992. PP. 205.

Abdellatif Bettayeb "The Geometric Least Squares Fitting Of Ellipses .IOSR Journal of Mathematics (IOSR-JM) 14.3 (2018): 12-18.