Journal of Interdisciplinary Mathematics
ISSN: 0972-0502 (Print), ISSN: 2169-012X (Online)
Vol. 27 (2024), No. 1, pp. 87–107
DOI : 10.47974/JIM-1727

A closed–form general solution for the distance of point-to-hyperbola in two dimensions

Chang-Chien Chou *
Gia-Shie Liu †
Fang-Ling Lin §
Department of Information Management
Lunghwa University of Science and Technology
Taiwan, R.O.C.

Abstract

We consider the simple problem, in a two dimensional plane, the shortest Euclidean distance from a point to a hyperbola. Get rid of existing analytical or approximating methods from the literature, we give a closed-form general equation to this problem. Adhered in the end of the paper, the C++ and Python codes for the resolved general solution can facilitate practitioners in all fields to conduct the solution to their applications immediately.

Subject Classification: *14Pxx: Real algebraic and real analytic geometry*

Keywords: *Distance to hyperbola, Distance to curve.*

1. Introduction

Hyperbola or hyperboloid reigned a variety of applications in our daily life: engineering, robotics, optics, acoustics, astronomy, …, and so on. For example, in civil engineering, hyperbolic curve and/or hyperboloid are used to calculate the maximum bearing capacity of a dam. A cooling tower of nuclear power plant, or a bridge, they all take hyperbola or hyperboloid during their engineering design. By drafting the hyperboloid to estimate

* *E-mail:* samuelchou7@yahoo.com.tw (Corresponding Author)
† *E-mail:* liug@gm.lhu.edu.tw
§ *E-mail:* fangling@gm.lhu.edu.tw

the bearing capacity is much easier than the complicated mathematical calculations. The most magical application can be thought of the time difference of arrival (TDOA) system. By sketching two hyperbolas' intersection, the FBI in the city or the captain on the sea can lock on their prey immediately even in the era before the birth of computer.

In spite of civil engineering, the astronomy has a broaden horizon taking advantage of hyperbola. The trajectory for the spaceship escaping from the gravity of earth towards the cosmos is nothing but a hyperbola when sitting down on earth as the origin of the coordinate. The Curiosity bounds for Mars, the Juno for Jupiter, they are voyaging along with their hyperbolic trajectories planned on earth.

Figure 1 illustrates the leanest form of a hyperbola $y(x)=\frac{1}{x}$, a rectangular hyperbola.

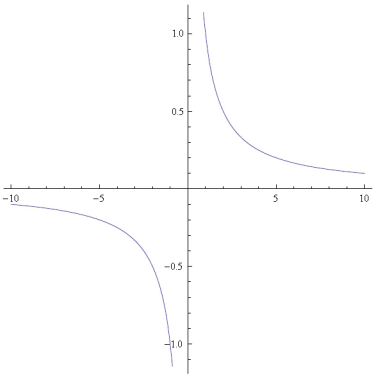

Figure 1

***y*(*x*) = 1/*x*, a rectangular hyperbola.**

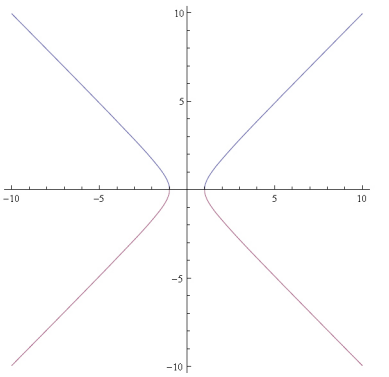

Figure 2

The unit hyperbola $x^2 - y^2 = 1$

By using the affine transformations, any hyperbola is the affine image of the unit hyperbola (Figure 2)

$$x^2 - y^2 = 1$$

In general, most popular representations of hyperbola can be either parametric (Eq. 1), or canonical (Eq. 2).

$$\begin{cases} x = a\sec t + h \\ y = b\tan t + k \end{cases} \text{ or } \begin{cases} x = a\cosh t + h \\ y = b\sinh t + k \end{cases} \tag{1}$$

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1 \tag{2}$$

Since hyperbola is a sibling of ellipse, means used in ellipse or parabola can probably also be useful in hyperbola. Chou has successfully developed closed-form solutions on the point-to-ellipse problem [1] as well as the point-to-parabola problem [2] in two dimensions. This paper is the last one of the series: point-to-ellipse, point-to-parabola, and point-to-hyperbola. This work takes similar philosophy resolving a closed-form general equation on the distance of one point to one hyperbola in two dimensions.

In MatLab©, there is a third party function called

Residuals_hyperbola(XY, ParG) which can calculate the approximate distance between one point (or points) to one hyperbola in two dimensions.

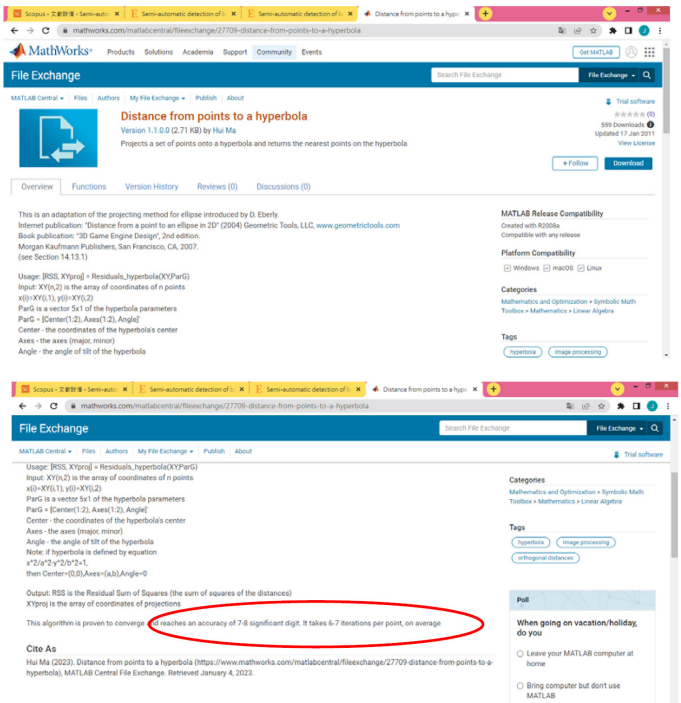

Figure 3

Calculating the distance between point and hyperbola is not that straightforward. Data Source: Hui Ma (2023). Distance from points to a hyperbola (https://www.mathworks.com/matlabcentral/fileexchange/27709-distance-from-points-to-a-hyperbola), MATLAB Central File Exchange. Retrieved January 8, 2023.

Yet unfortunately, the iterative means are taken into squeezing the distance as the function's description tells us in Figure 3. Once the requested precision increased, the convergent loops increases as well. It is neither an exact solution nor has a constant-time computation time, as we presented in this paper. Even such MatLab© does not give an efficient solution coping with this simple problem, there must be intrinsic complicated solution to be discovered.

2. Related Works

Versatile applications need to calculate the distance from points to a hyperbola: in ground penetrating radar [3–4], or in precise measurement [5–6]. Vesely and Doan (2015) [7] proposed an analytical algorithm to find the intersection points of two hyperbolas in 2D. Vesely's methods outperforms the existing numerical methods [8] in time consuming.

Uteshev (2018) [9] had thoughtfully studied the similar point-to-ellipse problem. They found, in the literature, existing archived approaches dealing with the point-to-ellipse problem can be analytical or numerical. The numerical methods try to generate loops coping with this constrained nonlinear optimization problem, that is, to create a succession of points on the border of the conics approaching to the shortest point (projecting the query point onto the conics) [10–12]. The analytical methods mostly come from the well–studied problem: the ellipse (ellipsoid) fitting [13–16]. When the coefficients of the quadric are unknown (not real numbers), the closeness from the query point to the shortest point on the quadric becomes knotty while in solving the ellipse fitting problem.

Nevertheless, all the above mentioned works [9–16] still did not give constant time response with actual values with respect to projecting the query point onto the conics. Means investigated in [9] are all kinds of approximating methods. Such analytical or numerical methods are time expensive. They are also getting stretched while implementing them into some industrial applications especially when the coefficients are unknown (for example, a moving objects toward a conic and vice versa). Iterative or approximating methods are time consuming as the increase of more precision digits acquired from industrial applications.

So far, from the literature, we found no efficient algorithm with O(1) time complexity and exact value to answer the question: what is the distance from a point to a hyperbola even in two dimensions. In the next section, we will illustrate how to derive such closed-form general solution

by taking some tips of algebraic substitution and fundamental geometry to reduce the complexity of the equation.

3. The General Solution

Consider point $P(x_p, y_p)$ be the query point towards the given hyperbola $\boldsymbol{\Gamma}$ in two dimensions. Let point $T(x_t, y_t)$ be the shortest point (the projecting point of P on $\boldsymbol{\Gamma}$) on the border of $\boldsymbol{\Gamma}$ with a minimum *Euclidean* distance such that line segment from P to T has the shortest length. Let $T'(x_{t'}, y_{t'})$ be the other projection on the other branch of the hyperbola. The distance from a query point $Q(u, v)$ to a conic $\boldsymbol{\Gamma}$ is the smallest *Euclidean* distance from Q to all the points in $\boldsymbol{\Gamma}$ [15]:

$$dist(Q, \Gamma) = \min\{\| (u,v) - (s,t) \| \mid (u,v) \in Q, (s.t) \in \Gamma\} \tag{3}$$

where $\|\bullet\|$ denotes the *Euclidean* distance.

The objective function Eq. (3) is a quartic equation with 4th order. The equation with degree four is the highest degree such that can be solved by radicals, for example, by using Ferrari's method (A method for reducing the solution of a quartic equation to the solution of one cubic plus two quadratic equations. Lodovico Ferrari published in 1545 [18]). However, our quartic equation is not that lean to directly adopt Ferrari's method. We take several algebraic substitutions to reduce the complexity of the equation. Symbols and variables used in the following derivations of equations are listed in Table 1.

Table 1

Symbols used in the derivations of equations.

$\boldsymbol{\Gamma}: \frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$	The hyperbola centered at (0, 0).
$P(x_p, y_p)$	The query point.
$T(x_t, y_t)$	The projection of P on Γ.
$T'(x_{t'}, y_{t'})$	The other projection of P on the other branch of Γ.
T_1, T_2, T_3, T_4	Four roots, two of them will be T and T'.
$\lvert\overline{PT}\rvert$	The length of the query line segment, from P to T.

WLOG, assume the hyperbola Γ be given through a canonical form

$$\Gamma: \frac{x^2}{a^2} - \frac{y^2}{b^2} = 1. \tag{4}$$

Through point T, the tangent line is thus

$$\frac{x_t x}{a^2} - \frac{y_t y}{b^2} = 1. \tag{5}$$

$\overleftrightarrow{PT}$ is perpendicular to that of Eq. (5), its slope is thus

$$\frac{y_t - y_p}{x_t - x_p} = -\frac{a^2 y_t}{b^2 x_t} \tag{6}$$

Taking $X = \frac{1}{x_t}$, $Y = \frac{1}{y_t}$ through Eqs. (4) to (6) we have

$$\frac{Y^2}{a^2} - \frac{X^2}{b^2} = X^2 Y^2 \tag{7}$$

and

$$Y = \frac{-a^2 x_p X + b^2 + a^2}{b^2 y_p} \tag{8}$$

Substituting Eq. (8) into Eq. (7) we obtain a quartic equation about X

$$-a^6 x_p^2 X^4 + 2a^4(a^2+b^2)x_p X^3 - (a^2 b^2 y_p^2 - a^4 x_p^2 + a^2(a^2+b^2)^2)X^2 - 2a^2(a^2+b^2)x_p X + (a^2+b^2)^2 = 0 \tag{9}$$

Solve Eq. (9) we come out four roots of X and Y as shown by Eq. (10).

$$X_1 = \frac{d}{2a^2 x_p} - \frac{k + \sqrt{n-l}}{2}$$

$$X_2 = \frac{d}{2a^2 x_p} - \frac{k - \sqrt{n-l}}{2}$$

$$X_3 = \frac{d}{2a^2 x_p} + \frac{k - \sqrt{n+l}}{2}$$

$$X_4 = \frac{d}{2a^2 x_p} + \frac{k + \sqrt{n+l}}{2} \tag{10}$$

$$Y_1 = \frac{-a^2 x_p X_1 + b^2 + a^2}{b^2 y_p}$$

$$Y_2 = \frac{-a^2 x_p X_2 + b^2 + a^2}{b^2 y_p}$$

$$Y_3 = \frac{-a^2 x_p X_3 + b^2 + a^2}{b^2 y_p}$$

$$Y_4 = \frac{-a^2 x_p X_4 + b^2 + a^2}{b^2 y_p}$$

where

$$c = -a^6 - 2a^4 b^2 - a^2 b^4 + a^4 x_p^2 - a^2 b^2 y_p^2,$$

$$d = a^2 + b^2,$$

$$e = a^4 + 2a^2 b^2 + b^4,$$

$$f = a^4 x_p + a^2 b^2 x_p,$$

$$g = e - a^2 x_p^2 + b^2 y_p^2,$$

$$h = -108a^6 f^2 x_p^2 + 108a^4 e f^2 + 72a^6 c e x_p^2 + 36a^2 c f^2 + 2c^3,$$

$$i = \sqrt[3]{h + \sqrt{h^2 + 4(12a^6 e x_p^2 - 12a^2 f^2 - c^2)^3}},$$

$$j = 3a^2 i x_p^2,$$

$$k = \sqrt{\frac{d^2 - g}{a^4 x_p^2} - \frac{c}{3a^6 x_p^2} - \frac{\sqrt[3]{2} g^2}{j} - \frac{i}{3\sqrt[3]{2} a^6 x_p^2}},$$

$$l = \frac{2d^3}{a^6 k x_p^3} - \frac{4d}{a^4 k x_p} - \frac{2dg}{a^6 k x_p^3},$$

$$m = \frac{i}{3\sqrt[3]{2} a^6 x_p^2},$$

$$n = \frac{2d^2 - g}{a^4 x_p^2} + \frac{c}{3a^6 x_p^2} + \frac{\sqrt[3]{2} g^2}{j} + m,$$

Bringing back $(X_1, Y_1) \sim (X_4, Y_4)$ to their original coordinates $T(x_t, y_t)$ by Eq. (11)

$$\begin{cases} x_{t1} = \dfrac{1}{X_1}, & y_{t1} = \dfrac{1}{Y_1} \\ x_{t2} = \dfrac{1}{X_2}, & y_{t2} = \dfrac{1}{Y_2} \\ x_{t3} = \dfrac{1}{X_3}, & y_{t3} = \dfrac{1}{Y_3} \\ x_{t4} = \dfrac{1}{X_4}, & y_{t4} = \dfrac{1}{Y_4} \end{cases} \tag{11}$$

Eq. (11) are roots of a quartic equation, there are two real roots and two complex conjugate roots in our point-to-hyperbola problem. We now calculate the distance between the two real roots and the query point P respectively, the shorter one is the answer for the point-to-hyperbola problem. The above manipulations can be denoted by Eq. (12).

$$dist(Q, \Gamma) = \min(\{|\overline{PT_1}|, |\overline{PT_2}|, |\overline{PT_3}|, |\overline{PT_4}|\}) \tag{12}$$

Below we illustrate an example to empirically shown the correctness and effectiveness of the proposed closed-form equation. WLOG, assume the given hyperbola Γ is

$$\Gamma : \frac{x^2}{a^2} - \frac{y^2}{b^2} = 1 \text{ where } a = 2,\ b = 5.$$

The query point P(xp, yp) = P(10, 12) as shown in Figure 4.

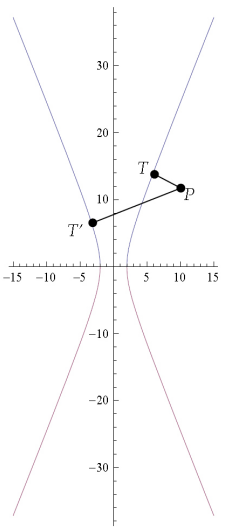

Figure 4

The point-to-hyperbola problem.

By applying Eqs. (10) and (11), we result in four points

$$\begin{cases} T_1(x_{t1}, y_{t1}) = (-3.598317613520,\ 7.478255832651) \\ T_2(x_{t2}, y_{t2}) = (0.28408135107668 + 0.53351498941133\ i, \\ \qquad\qquad -0.1846798941356 - 5.1292013852211\ i) \\ T_3(x_{t3}, y_{t3}) = (5.788775601022,\ 13.580759128034) \\ T_4(x_{t4}, y_{t4}) = (0.28408135107668 - 0.53351498941133\ i, \\ \qquad\qquad -0.1846798941356 + 5.1292013852211\ i) \end{cases}$$

with their corresponding length

$$\begin{cases} |\overline{PT_1}| = 14.330401677313 \\ |\overline{PT_2}| = 15.1829512098539 + 3.7748977760778\ i \\ |\overline{PT_3}| = 4.498134097535 \\ |\overline{PT_4}| = 15.182951209854 - 3.774897776078\ i \end{cases}$$

Lastly, Eq. (12) disregards the complex number results and then go to choose T_3

$$T_3(x_{t3}, y_{t3}) = (5.788775601022,\ \ 13.580759128034)$$

which generates the shortest distance

$$|\overline{PT_3}| = 4.498134097535.$$

Please note, all symbols used through Eqs. (4) to (12) must be declared with complex numbers when programming. Since we use radicals by radicals in our equations, the complex number data type is a must. Now we further launch a couple of trials to test the correctness of Eq. (12) intensively. We evenly spread 100 equal points on the contour of the hyperbola by the "for loop"

for $(k = 1,\ k \le 100,\ k++)$

$$T_i\left(a\ Sec\left(\frac{k}{100}\ 2\pi\right),\ b\ Tan\left(\frac{k}{100}\ 2\pi\right)\right);$$

Hence, we can calculate each length of line segment $|\overline{PT_i}|$.

Table 2 and Figure 5 together gave the experimental results, where i denotes the number of iteration. θ depicts the angle used in $T_i(a\ \mathrm{Sec}\theta,\ b\ \mathrm{Tan}\theta)$ where $\theta = 2\pi\frac{i}{100}$. $|\overline{PT_i}|$ represents the distance between T_i and P.

Table 2

100 points testing data on the border of the hyperbola.

i	θ	$\lvert\overline{PT_i}\rvert$	i	θ	$\lvert\overline{PT_i}\rvert$
1	0.062831853071796	14.1593058591172	51	3.20442450666159	16.7524361290447
2	0.125663706143592	13.8919174341771	52	3.26725635973338	16.5414995590707
3	0.188495559215388	13.6177378442197	53	3.33008821280518	16.3366262076714
4	0.251327412287183	13.3343013663359	54	3.39292006587698	16.1368665682547
5	0.314159265358979	13.0388940843740	55	3.45575191894877	15.9414471389076
6	0.376991118430775	12.7284546320340	56	3.51858377202057	15.7497854550952
7	0.439822971502571	12.3994529977780	57	3.58141562509236	15.5615269477200
8	0.50265482457437	12.0477382418360	58	3.64424747816416	15.3766133147445
9	0.56548667764616	11.6683432515877	59	3.70707933123596	15.1953989817914
10	0.62831853071796	11.2552314433945	60	3.76991118430775	15.0188439316866
11	0.69115038378975	10.8009677668769	61	3.83274303737955	14.8488317007060
12	0.75398223686155	10.2962986885793	62	3.89557489045134	14.6886983265922
13	0.81681408993335	9.7296469366696	63	3.95840674352314	14.5441265616758
14	0.87964594300514	9.0866125292370	64	4.02123859659494	14.4246902059253
15	0.94247779607694	8.3498798443028	65	4.08407044966673	14.3465892375373
16	1.00530964914873	7.5010483509585	66	4.14690230273853	14.3376324300584
17	1.06814150222053	6.5302513425051	67	4.20973415581032	14.4465965268015
18	1.13097335529233	**5.4779640980868**	68	4.27256600888212	14.7613977368092
19	1.19380520836412	**4.6101071796435**	69	4.33539786195391	15.4457385560248
20	1.25663706143592	**4.8915436350876**	70	4.39822971502571	16.8170341420225
21	1.31946891450771	7.7259046944786	71	4.46106156809751	19.5288346303164
22	1.38230076757951	14.2268649399439	72	4.52389342116930	25.0866617747918
23	1.44513262065130	28.215186088987	73	4.58672527424110	37.873408963030
24	1.50796447372310	70.923027941251	74	4.64955712731289	79.398700122535
25	1.57079632679490	**Infinity**	75	4.71238898038469	**Infinity**
26	1.63362817986669	100.592466634813	76	4.77522083345649	94.046619582435
27	1.69646003293849	57.742451694350	77	4.83805268652828	51.921983275194
28	1.75929188601028	43.444961255248	78	4.90088453960008	38.216851590451
29	1.82212373908208	36.2782809295233	79	4.96371639267187	31.5345507107892
30	1.88495559215388	31.9602360169009	80	5.0265482457437	27.6146926120141
31	1.94778744522567	29.0644395268242	81	5.0893800988155	25.0484309623408
32	2.01061929829747	26.9800819995419	82	5.1522119518873	23.2386327527602

Contd...

33	2.07345115136926	25.4020182378418	83	5.2150438049591	21.8906957848248
34	2.13628300444106	24.1608607067699	84	5.2778756580309	20.8433493651067
35	2.19911485751286	23.1550487726479	85	5.3407075111026	20.0013038156860
36	2.26194671058465	22.3199877158394	86	5.4035393641744	19.3048359577281
37	2.32477856365645	21.6126273072016	87	5.4663712172462	18.7147020064492
38	2.38761041672824	21.0031630933493	88	5.5292030703180	18.2040865763882
39	2.45044226980004	20.4702972599151	89	5.5920349233898	17.7540466298393
40	2.51327412287183	19.9983969298953	90	5.6548667764616	17.3508052137549
41	2.57610597594363	19.5757191854200	91	5.7176986295334	16.9840768264710
42	2.63893782901543	19.1932632336829	92	5.7805304826052	16.6459939105499
43	2.70176968208722	18.8440056366888	93	5.8433623356770	16.3303968762258
44	2.76460153515902	18.5223773850168	94	5.9061941887488	16.0323510263218
45	2.82743338823081	18.2238981554025	95	5.9690260418206	15.7478089282103
46	2.89026524130261	17.9449154116120	96	6.0318578948924	15.4733680695743
47	2.95309709437441	17.6824151173315	97	6.0946897479642	15.2060919544224
48	3.01592894744620	17.4338824846849	98	6.1575216010360	14.9433738099888
49	3.07876080051800	17.1971985043668	99	6.2203534541078	14.6828288334675
50	3.14159265358979	16.9705627484771	100	6.2831853071796	14.4222051018560

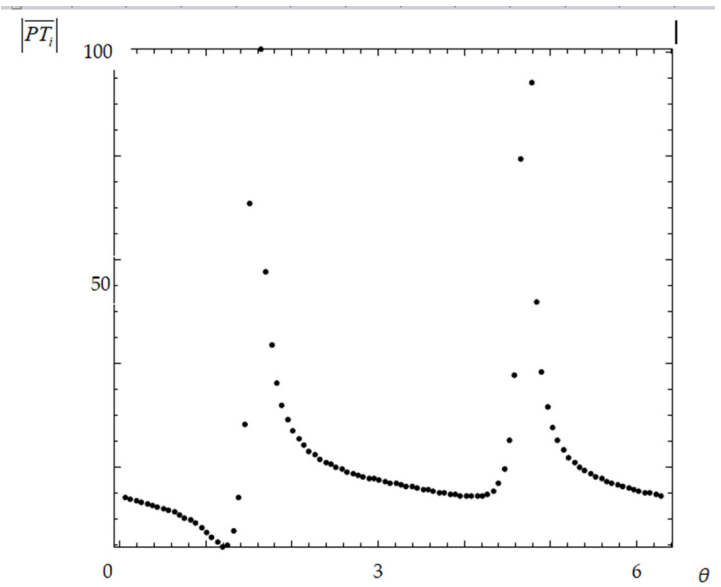

Figure 5

100 lengths of $|\overline{PT_i}|$ varies by θ.

Figure 5 shows the lengths of $|\overline{PT_i}|$ varied by θ. Observing that, in Table 2, the minimum path happens only in between iteration 18 and 20, we launch another round of 100 deeper points tests between the angle from iteration 18 to 20 to further deepen down the optimal point to be shown in Table 3 and Figure 6.

Table 3

Another 100 deeper points testing data on the border of the hyperbola.

i	θ	$\lvert\overline{PT_i}\rvert$	i	θ	$\lvert\overline{PT_i}\rvert$
1	1.13222999235376	5.4571510062228	51	1.19506184542556	4.5995626605077
2	1.13348662941520	5.4363918150720	52	1.19631848248699	4.5894595677992
3	1.13474326647663	5.4156902756913	53	1.19757511954843	4.5798106191725
4	1.13599990353807	5.3950502544506	54	1.19883175660987	4.5706287088087
5	1.13725654059951	5.3744757360281	55	1.20008839367130	4.5619269008722
6	1.13851317766094	5.3539708264527	56	1.20134503073274	4.5537184223749
7	1.13976981472238	5.3335397561908	57	1.20260166779417	4.5460166554388
8	1.14102645178381	5.3131868832769	58	1.20385830485561	4.5388351289564
9	1.14228308884525	5.2929166964846	59	1.20511494191704	4.5321875096467
10	1.14353972590668	5.2727338185346	60	1.20637157897848	4.5260875925113
11	1.14479636296812	5.2526430093375	61	1.20762821603992	4.5205492906962
12	1.14605300002956	5.2326491692678	62	1.20888485310135	4.5155866247693
13	1.14730963709099	5.2127573424636	63	1.21014149016279	4.5112137114297
14	1.14856627415243	5.1929727201500	64	1.21139812722422	4.5074447516644
15	1.14982291121386	5.1733006439794	65	1.21265476428566	4.5042940183769
16	1.15107954827530	5.1537466093842	66	1.21391140134710	4.5017758435135
17	1.15233618533674	5.1343162689358	67	1.21516803840853	4.4999046047197
18	1.15359282239817	5.1150154357047	68	1.21642467546997	**4.4986947115608**
19	1.15484945945961	5.0958500866125	69	1.21768131253140	**4.4981605913486**
20	1.15610609652104	5.0768263657713	70	1.21893794959284	**4.4983166746175**
21	1.15736273358248	5.0579505878002	71	1.22019458665428	4.4991773802998
22	1.15861937064392	5.0392292411108	72	1.22145122371571	4.5007571006529
23	1.15987600770535	5.0206689911540	73	1.22270786077715	4.5030701859956
24	1.16113264476679	5.0022766836152	74	1.22396449783858	4.5061309293139
25	1.16238928182822	4.9840593475504	75	1.22522113490002	4.5099535508004
26	1.16364591888966	4.9660241984499	76	1.22647777196146	4.5145521823955

Contd...

27	1.16490255595110	4.9481786412177	77	1.22773440902289	4.5199408523978
28	1.16615919301253	4.9305302730557	78	1.22899104608433	4.5261334702181
29	1.16741583007397	4.9130868862362	79	1.23024768314576	4.5331438113479
30	1.16867246713540	4.8958564707510	80	1.23150432020720	4.5409855026191
31	1.16992910419684	4.8788472168213	81	1.23276095726863	4.5496720078276
32	1.17118574125827	4.8620675172524	82	1.23401759433007	4.5592166137974
33	1.17244237831971	4.8455259696170	83	1.23527423139151	4.5696324169577
34	1.17369901538115	4.8292313782495	84	1.23653086845294	4.5809323105060
35	1.17495565244258	4.8131927560349	85	1.23778750551438	4.5931289722278
36	1.17621228950402	4.7974193259722	86	1.23904414257581	4.6062348530402
37	1.17746892656545	4.7819205224933	87	1.24030077963725	4.6202621663236
38	1.17872556362689	4.7667059925186	88	1.24155741669869	4.6352228781014
39	1.17998220068833	4.7517855962293	89	1.24281405376012	4.6511286981256
40	1.18123883774976	4.7371694075334	90	1.24407069082156	4.6679910719150
41	1.18249547481120	4.7228677142074	91	1.24532732788299	4.6858211737962
42	1.18375211187263	4.7088910176904	92	1.24658396494443	4.7046299009817
43	1.18500874893407	4.6952500325094	93	1.24784060200587	4.7244278687222
44	1.18626538599551	4.6819556853151	94	1.24909723906730	4.7452254065569
45	1.18752202305694	4.6690191135053	95	1.25035387612874	4.7670325556825
46	1.18877866011838	4.6564516634158	96	1.25161051319017	4.7898590674540
47	1.19003529717981	4.6442648880576	97	1.25286715025161	4.8137144030221
48	1.19129193424125	4.6324705443793	98	1.25412378731305	4.8386077341071
49	1.19254857130269	4.6210805900356	99	1.25538042437448	4.8645479449019
50	1.19380520836412	4.6101071796435	100	1.25663706143592	4.8915436350876

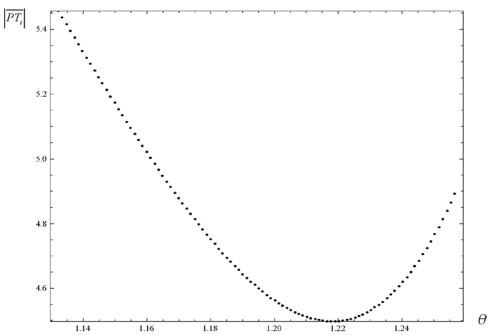

Figure 6

Variation of lengths $\overline{|PT_i|}$ by θ from another 100 deeper points.

The above two consecutive experiments all together again have confirmed the correctness of the proposed point-to-hyperbola closed-form general equation empirically. Adhered in the appendix, practitioners in many fields will soon find the merits on the source code based on Eqs. (4) through (11). The codes are programmed by C++ and Python under Microsoft Windows® operating system. Those two versions of different programming languages codes lead to identical results.

4. Conclusions

A closed-form general equation is the best answer for the practitioners. For example, engineers can compute the peak of a parabolic moving object immediately by applying the equation

$$\frac{-b\pm\sqrt{b^2-4ac}}{2a}.$$

Likewise, in this work, the derived closed-form general equation can instantly answer the question: what is the precise (as you wish how many decimal digits) distance from a point to a hyperbola in two dimensions. By using fundamental algebraic geometry with some algebraic substitution tips, the proposed closed-form equations derived four roots to the point-to-hyperbola problem. Among the four roots, we can discard the two complex conjugate roots. Then we compute the two real roots. The minimum one is the answer. Henceforth, we no longer need the existing analytical or numerical methods from the literature [9]. The distance form one point to one hyperbola is completely settled by the closed-form equation we presented in this work.

Many algorithms can be taking advantage of the closed-form equation proposed in this paper since. For instance, if we can come out a general equation for the shortest distance in a point-hyperbola-point touring problem, then we have the chance deriving an exact or approximate exact algorithms computing the shortest path touring a plural number of hyperbolas. Since this is a closed-form general equation, the calculation results can be obtained by constant time O(1), such that it can be implemented to any computer systems directly to fulfill the industrial application needs. Practitioners will soon find the merits from the results of this work.

On the applications of discrete finite fields [19–22], there also have a core calculation need about distance to hyperbola that the program codes adhered in the appendix of this paper can immediately contribute then.

Appendix A: C++ source code for the point-to-hyperbola.

```
// PH_C++.cpp : Powered by Dr. Chou
// samuelchou7@yahoo.com.tw
//#include "stdafx.h" //include it when running Microsoft Windows VS C++
#include <complex>
#include <iostream>
#include <iomanip>
using namespace std;
// Input: a, b for the axes of the hyperbola;
// Input: xp, yp for the axes of query point P(xp, yp)
// Output: PT1, PT2, PT3, and PT4. Choose the shortest one.
int main()
{
    complex<double> a, b, xp, yp;
    complex<double> c, d, e, f, g, h, i, j, k, l, m, n, xt1, xt2, xt3, xt4, yt1, yt2, yt3, yt4, X1, X2, X3, X4, Y1, Y2, Y3, Y4, PT1, PT2, PT3, PT4;
    complex<double> c1, c2, c3, c4, c6, c8, c12, c16, c36, c72, c108;
    complex<double> a2, a4, a6, b2, b4;
    c1 = 1.0; c2 = 2.0; c3 = 3.0; c4 = 4.0; c6 = 6.0; c8 = 8.0; c12 = 12.0; c16 = 16.0; c36 = 36.0; c72 = 72.0; c108 = 108.0;
    a = 2.0; b = 5.0; xp = 10.0; yp = 12.0;
    a2 = a*a; a4 = pow(a, c4); a6 = pow(a, c6); b2 = b*b; b4 = pow(b, c4);
    c = -a6 - c2 * a4*b2 - a2*b4 + a4*xp*xp - a2*b2*yp*yp;
    d = a2 + b2;
    e = a4 + c2 *a2*b2 + b4;
    f = a4*xp + a2*b2*xp;
    g = e - a2*xp*xp + b2*yp*yp;
    h = -c108 * a6*xp*xp*f*f + c108 * e *a4*f*f + c72* a6*e*xp*xp*c + c36 *f*f*a2*c + c2 * c*c*c;
    i = pow(h + sqrt(-c4 * pow(((-c12*a6*e*xp*xp) + (c12*f*f*a2) + (c*c)), c3) + (h*h)), (c1 / c3));
    j = c3 *a2*xp*xp*i;
    k = sqrt(((d*d - g) / (a4*xp*xp)) - (c / (c3*a6*xp*xp)) - ((pow(c2, (c1 / c3))*g*g) / j) - (i / (c3*pow(c2, (c1 / c3))*a6*xp*xp)));
```

```
    l = ((c8*d*d*d / (a6*xp*xp*xp)) - ((c16*d) / (a4*xp)) - ((c8 *d*g) /
    (a6*xp*xp*xp))) / (c4 * k);
    m = i / (c3*pow(c2, (c1 / c3))*a6*xp*xp);
    n = ((c2*d*d - g) / (a4*xp*xp)) + (c / (c3*a6*xp*xp)) + ((pow(c2, (c1 /
    c3))*g*g) / j) + m;

    X1 = (d / (c2*a2*xp)) - (k / c2) - ((sqrt(n - l)) / c2);
    X2 = (d / (c2*a2*xp)) - (k / c2) + ((sqrt(n - l)) / c2);
    X3 = (d / (c2*a2*xp)) + (k / c2) - ((sqrt(n + l)) / c2);
    X4 = (d / (c2*a2*xp)) + (k / c2) + ((sqrt(n + l)) / c2);
    Y1 = (-X1*xp*a2 + b2 + a2) / (yp*b2);
    Y2 = (-X2*xp*a2 + b2 + a2) / (yp*b2);
    Y3 = (-X3*xp*a2 + b2 + a2) / (yp*b2);
    Y4 = (-X4*xp*a2 + b2 + a2) / (yp*b2);
    xt1 = c1 / X1; yt1 = c1 / Y1;
    xt2 = c1 / X2; yt2 = c1 / Y2;
    xt3 = c1 / X3; yt3 = c1 / Y3;
    xt4 = c1 / X4; yt4 = c1 / Y4;
    PT1 = sqrt(pow(xt1 - xp, c2) + pow(yt1 - yp, c2));
    PT2 = sqrt(pow(xt2 - xp, c2) + pow(yt2 - yp, c2));
    PT3 = sqrt(pow(xt3 - xp, c2) + pow(yt3 - yp, c2));
    PT4 = sqrt(pow(xt4 - xp, c2) + pow(yt4 - yp, c2));
    cout << "a=" << a << " b=" << b << " (xp, yp)= " << xp << ", " << yp
<< endl;
    cout << "xt1=" << setprecision(15) << xt1 << endl << "yt1=" << yt1
<< endl << "xt2=" << xt2 << endl << "yt2=" << yt2 << endl << "xt3="
<< xt3 << endl << "yt3=" << yt3 << endl;
    cout << "xt4=" << xt4 << endl << "yt4=" << yt4 << endl << "PT1="
<< PT1 << endl << "PT2=" << PT2 << endl << "PT3=" << PT3 << endl
<< "PT4=" << PT4 << endl;
    //Choose the minimum value from among PT1 to PT4
    cin.get();
    return 0;
}
```

Print screen running PH_C++.exe is shown by Figure 7.

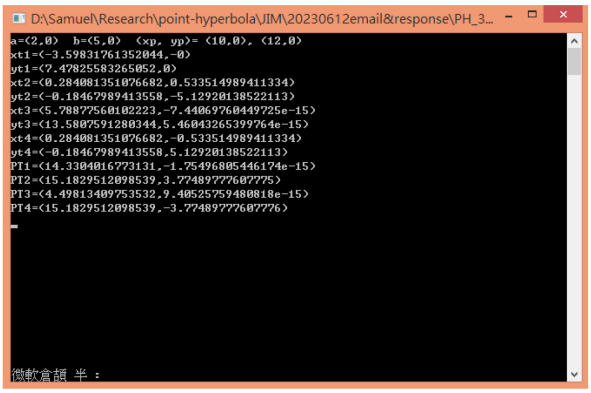

Figure 7

Results of running PH_C++.exe.

Appendix B: Python source code of point-to-hyperbola PH_.py.

```
# Powered by Dr. CC Chou
# samuelchou7@yahoo.com.tw
import cmath
c1 = 1.0; tWo = 2.0; three = 3.0; four = 4.0; six = 6.0; eight = 8.0; ten = 10.0; twelve = 12.0; oneSix = 16.0; threeSix = 36.0; sevenTwo = 72.0; c108 = 108.0;
a = 2.0; b = 5.0; xp = 10.0; yp = 12.0;
a2=a*a
a4=a*a*a*a
a6=a*a*a*a*a*a
b2=b*b
b4=b*b*b*b
c = -a6 - tWo * a4*b2 - a2*b4 + a4*xp*xp - a2*b2*yp*yp
d = a2 + b2
e = a4 + tWo *a2*b2 + b4
f = a4*xp + a2*b2*xp
g = e - a2*xp*xp + b2*yp*yp
h = -c108 * a6*xp*xp*f*f + c108 * e *a4*f*f + sevenTwo * a6*e*xp*xp*c + threeSix *f*f*a2*c + tWo * c*c*c
i=pow(h+cmath.sqrt(-four*pow(((-twelve*a6*e*xp*xp)+(twelve*f*f*a2) + (c*c)), three) + (h*h)), (c1 / three))
j = three *a2*xp*xp*i
```

```
k = cmath.sqrt(((d*d-g)/(a4*xp*xp))-(c/(three*a6*xp*xp))-
((pow(tWo,(c1/three))*g*g)/j)-(i/(three*pow(tWo,(c1/
three))*a6*xp*xp)))
l = ((eight*d*d*d / (a6*xp*xp*xp)) - ((oneSix*d) / (a4*xp)) - ((eight *d*g)
/ (a6*xp*xp*xp))) / (four * k)
m = i / (three*pow(tWo, (c1 / three))*a6*xp*xp)
n = ((tWo*d*d - g) / (a4*xp*xp)) + (c / (three*a6*xp*xp)) + ((pow(tWo,
(c1 / three))*g*g) / j) + m
X1 = (d / (tWo*a2*xp)) - (k / tWo) - ((cmath.sqrt(n - l)) / tWo)
X2 = (d / (tWo*a2*xp)) - (k / tWo) + ((cmath.sqrt(n - l)) / tWo)
X3 = (d / (tWo*a2*xp)) + (k / tWo) - ((cmath.sqrt(n + l)) / tWo)
X4 = (d / (tWo*a2*xp)) + (k / tWo) + ((cmath.sqrt(n + l)) / tWo)
Y1 = (-X1*xp*a2 + b2 + a2) / (yp*b2)
Y2 = (-X2*xp*a2 + b2 + a2) / (yp*b2)
Y3 = (-X3*xp*a2 + b2 + a2) / (yp*b2)
Y4 = (-X4*xp*a2 + b2 + a2) / (yp*b2)

tx1 = c1 / X1
ty1 = c1 / Y1
tx2 = c1 / X2
ty2 = c1 / Y2
tx3 = c1 / X3
ty3 = c1 / Y3
tx4 = c1 / X4
ty4 = c1 / Y4
PT1 = cmath.sqrt(pow(tx1 - xp, tWo) + pow(ty1 - yp, tWo))
PT2 = cmath.sqrt(pow(tx2 - xp, tWo) + pow(ty2 - yp, tWo))
PT3 = cmath.sqrt(pow(tx3 - xp, tWo) + pow(ty3 - yp, tWo))
PT4 = cmath.sqrt(pow(tx4 - xp, tWo) + pow(ty4 - yp, tWo))

print("a=", a, " b=", b," (xp, yp)= ", xp, ", ", yp, "\n")
print("xt1=", format(tx1, ".15f"), "\n")
print("yt1=", format(ty1, ".15f"), "\n")
print("xt2=", format(tx2, ".15f"), "\n")
print("yt2=", format(ty2, ".15f"), "\n")
print("xt3=", format(tx3, ".15f"), "\n")
```

```
print("yt3=", format(ty3, ".15f"), "\n")
print("xt4=", format(tx4, ".15f"), "\n")
print("yt4=", format(ty4, ".15f"), "\n")
print("PT1=", format(PT1, ".15f"), "\n")
print("PT2=", format(PT2, ".15f"), "\n")
print("PT3=", format(PT3, ".15f"), "\n")
print("PT4=", format(PT4, ".15f"), "\n")
```

Print screen running PH_.py is shown by Figure 8.

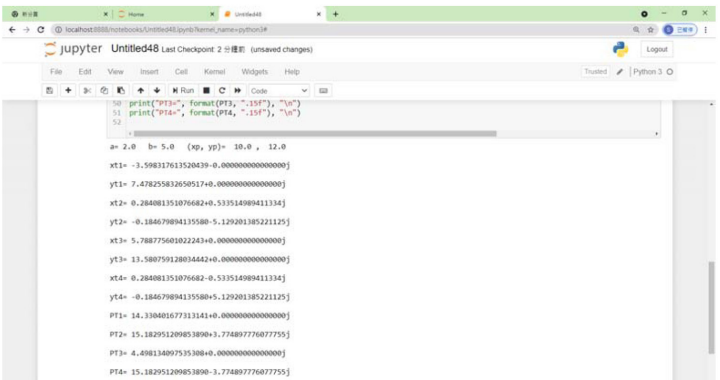

Figure 8

Results of running PH_.py.

Acknowledgement

This work is supported by the National Science and Technology Council, Taiwan, R.O.C., under the grant NSTC 112-2221-E-262 -005 -.

References

[1] Chou, C.-C. A closed-form general solution for the distance of point-to-ellipse in two dimensions. *Journal of Interdisciplinary Mathematics*. 22(3), 337–351 (2019).

[2] Chou, C.-C. A closed-form general solution for the distance of point-to-parabola in two dimensions. *International Journal of Computing Science and Mathematics*. 15(2), 132–141 (2022).

[3] Lei, W.; Hou, F.; Xi, J.; Tan, Q.; Xu, M.; Jiang, X.; Liu G.; Gu, Q. Automatic hyperbola detection and fitting in GPR B-scan image. *Automation in Construction*. 106, 102839 (2019).

[4] Wang, Y.; Cui, G.; Xu, J. Semi-automatic detection of buried rebar in GPR data using a genetic algorithm. *Automation in Construction*. 114, 103186 (2020).

[5] Tu, X.; Lei, X.; Ma, W.; Zuo, X.; Man, R. Hyperbola minimum-zone evaluation and fitting based on geometry optimised search algorithm. *Measurement*, 127, 205–209 (2018).

[6] Dai, Q.; Li, Y.; Jiang, T.; Sun, Q. A New B-Scan Interpretation Model in Complex Environments. *IEEE Transactions on Instrumentation and Measurement*. 71, 1–15 (2022).

[7] Vesely, J.; Van Doan, S. Analytical method solving system of hyperbolic equations. In Proceedings of the 2015 25th International Conference Radioelektronika. 21-22 April 2015, Pardubice, Czech Republic.

[8] Chan, Y.T.; Ho, K.C. A simple and efficient estimator for hyperbolic location. *IEEE Transactions on Image Processing*. 42(8), 1905–1915 (1994).

[9] Uteshev, A.Y.; Goncharova, M.V. Point-to-ellipse and point-to-ellipsoid distance equation analysis. *Journal of Computational and Applied Mathematics*. 328(15), 232–251 (2018).

[10] Dai, Y.H. Fast algorithms for projection on an ellipsoid. *SIAM Journal on Optimization*. 16(4), 986–1006 (2006).

[11] Chernov, N.; Wijewickrema, S. Algorithms for projecting points onto conics. *Journal of Computational and Applied Mathematics*. 251, 8–21 (2013).

[12] Tamasyan, G. Sh.; Chumakov, A.A. Finding the distance between ellipsoids. *Journal of Applied and Industrial Mathematics*. 8(3), 400–410 (2014).

[13] Sampson, P.D. Fitting conic sections to very scattered data: an iterative refinement of the Bookstein algorithm. *Computer Graphics and Image Processing*. 18, 97–108 (1982).

[14] Taubin, G. Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations with application to edge and range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 13(11), 1115–1138 (1991).

[15] Fitzgibbon, A.W.; Pilu, M.; Fisher, R.B. Direct least-squares fitting of ellipses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 21(5), 476–480 (1999).

[16] Szpak, Z.; Chojnacki, W.; van den Hengel, A. Guaranteed ellipse fitting with the Sampson Distance. in: LNCS. , 7576, 87–100 (2012).

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

[18] Smith, D.E. History of Mathematics, New York: Dover Publications, Vol 1, pp. 300 (1958).

[19] Rudnev, M.; Wheeler , J. On incidence bounds with Möbius hyperbolae in positive characteristic. *Finite Fields and Their Applications*. 78, 101978 (2022).

[20] Mohammadi, A.; Pham, T.; Warren, A. 10. A point-conic incidence bound and applications over Fp. *European Journal of Combinatorics.* 107, 103596 (2023).

[21] Maalek R.; Lichti, D.D. New confocal hyperbola-based ellipse fitting with applications to estimating parameters of mechanical pipes from point clouds. *Pattern Recognition*. 116, 107948 (2021).

[22] Dogan, I.; Akpinar, A. On Distance in Some Finite Planes and Graphs Arising from Those Planes. *Journal of Mathematics*, Article ID 6668682, 17 pages (2021).

Received January, 2023