Санкт-Петербургский государственный университет
Кафедра технологии программирования

Савченко Владислав Андреевич

Выпускная квалификационная работа бакалавра

Сравнение некоторых методов определения расстояния между объектами

Направление 010400

Прикладная математика и информатика

Научный руководитель,
кандидат физ.-мат. наук,
доцент
Гончарова М.В.

Санкт-Петербург
2018

Содержание

Задача исследования.

Подбор кривых и поверхностей второго порядка, в том числе подбор эллипсоидов, является довольно полезным инструментом и имеет множество практических приложений в различных прикладных областях, например, находит довольно широкое применение в задачах компьютерного зрения, таких как распознавание лиц [1], определение форм объектов [2], калибровка камеры [3] и другие.

Исследования в данной области можно разделить на два направления: первое сосредоточено на исследовании методов, с помощью которых можно получить оценку параметров кривой на основе некоторых эмпирических данных, например, применение различных методов оптимизации, а также робастной статистики для уменьшения влияния выбросов.

Второе направление сконцентрировано на поиске новых методов нахождения приближенного расстояния от точек до кривых или поверхностей. Задача нахождения точного расстояния является ресурсоемкой для прикладных задач [4], поэтому в данном направлении исследователи ищут наименее затратные приближенные методы, обеспечивающие требуемую точность результата. Существует два подхода к решению задачи: численный (итерационный) и аналитический (символьный). Решение задачи аналитически в общем случае требует нахождение наименьшего положительного корня уравнения, которое называют уравнением расстояний.

Одной из главных проблем данного направления можно назвать тот факт, что методы, разрабатываемые исследователями, не являются универсальными и, соответственно, имеют довольно узкую сферу применения. Так, многие из известных методов [5, 6] применимы только в двумерном случае и не могут быть обобщены на пространства большей размерности. Таким образом, возможность выбора методов для проведения расчетов в трехмерном пространстве и пространствах большей размерности является весьма ограниченной. Кроме того, многие методы имеют определенные недостатки (малая точность, наличие серьезных выбросов), которые могут быть недопустимыми для некоторых приложений. Учитывая всё вышеизложенное, поиск новых универсальных методов нахождения приближенного расстояния и их оценка, качественное сравнение с другими существующими решениями являются актуальными задачами в данной области.

В ходе работы над проблемой нахождения расстояния от точки до эллипса и от точки до эллипсоида А.Ю.Утешевым и М.В.Гончаровой была предложена формула для нахождения расстояния от точки до квадрики в форме (3). Была поставлена задача разработать программное обеспечение для проведения компьютерного тестирования данной формулы, чтобы сравнить её с другими уже имеющимися аналогами, выявить качественные характеристики данной формулы и решить вопрос о пригодности её применения в практических задачах.

1 Обзор приближенных формул нахождения расстояния.

Пусть дано уравнение эллипсоида в форме

$$G(X) = X^T AX + 2B^T X - 1 = 0,$$

здесь

$A -$ вещественная, симметричная, знакоопределенная матрица квадратичной формы размерности 3x3,

$B -$ заданный столбец размерности 3x1,

$X -$ столбец переменных размерности 3x1,

и дана точка

$$X_0 = (x_1, x_2, x_3)^T.$$

Необходимо найти *приближенное* расстояние от точки до эллипсоида.

К решению данной задачи можно подойти с помощью разных подходов. Например, в случае метода [7] (Sampson's distance) предлагается линеаризовать исходную функцию в окрестности X_0 и принять за приближение расстояние от точки до полученной линеаризации.

В альтернативном подходе предлагается ввести расстояние как новую переменную системы для нахождения точного расстояния

$$z = \min (X - X_0)^T (X - X_0)$$

и получить алгебраическое уравнение относительно z - *уравнение расстояния*

$$a_4 z^4 + a_3 z^3 + a_2 z^2 + a_1 z + a_0 = 0. \quad (1)$$

Наименьший положительный некратный корень данного уравнения - квадрат расстояния от точки до эллипсоида. Оценку данного корня также можно получить разными методами. Так, в работе [8] предлагается разложить данный корень в ряд по $G(X)$

$$z_0 = l_1 G(X_0) + l_2 G^2(X_0) + l_3 G^3(X_0) \ldots$$

В работе [9] авторы предлагают выразить данный корень через коэффициенты уравнения (1)

$$Q(a_n, \ldots, a_0) \approx z_0.$$

Также существует ряд методов, которые разработаны с учетом особенностей кривых второго порядка на плоскости [5, 6] и не могут быть обобщены на случаи пространств большей размерности.

Темой данной работы является сравнение метода, предложенного А.Ю.Утешевым и М.В.Гончаровой [8] с двумя методами, которые уже успели хорошо себя зарекомендовать в данной области. Первый из них - уже упомянутое выше расстояние Сампсона, второй - метод, предложенный M. Harker and P. O'Leary [9]. Данные методы были выбраны также потому, что могут быть использованы в пространствах любой размерности. Большинство работ по данной тематике посвящено сравнению различных методов в двумерном пространстве, однако интерес представляет и сравнение в пространствах большей размерности, так как это значительно расширяет сферу применимости исследуемой формулы: 3D моделирование, вычисление расстояния между объектом и множеством в некотором параметрическом пространстве и т.д.

Рассмотрим более подробно методы, выбранные для тестирования. Пусть дано уравнение квадрики, записанное в следующем виде

$$\mathrm{G(X)} = \mathrm{X}^{\mathrm{T}}\mathrm{AX} + 2\mathrm{B}^{\mathrm{T}}\mathrm{X} - 1 = 0,$$

чтобы найти расстояние от точки до квадрики, линеаризуем квадрику в окрестности точки X_0 посредством разложения функции в ряд Тейлора

$$\tilde{G}(X) = G(X_0) + \nabla G(X_0)^T (X - X_0).$$

Приближение $\tilde{G}(X)$ - прямая в двумерном случае и плоскость в трехмерном, которая, однако, не является касательной (касательной плоскостью) к исходной функции, если только X_0 не принадлежит $G(X)$. Расстояния от точки X_0 до $\tilde{G}(X)$

$$d_s = \frac{|G(X_0)|}{2\sqrt{(AX_0 + B)^T (AX_0 + B)}}. \qquad (2)$$

Данное выражение можно принять за приближенное расстояние от точки до эллипса/эллипсоида (Sampson's distance)[7]

В результате развития исследования [8] была получена формула

$$d_4 = d_s \sqrt{1 + \frac{k}{2||\nabla G||^4} G(X_0) + \frac{k^2}{2||\nabla G||^8} G(X_0)^2 + \frac{5}{8}\frac{k^3}{||\nabla G||^{12}}} \qquad (3),$$

где

$\nabla G -$ градиент функции G в точке X_0,

$\|\nabla G\| -$ евклидова норма градиента в точке X_0,

$k = (\nabla G)^T Hess(G)(\nabla G),\ Hess(G) -$ матрица Гессе.

M. Harker and P. O'Leary предложили использовать следующее выражение для вычисления приближенного расстояния от точки до эллипсоида

$$d_{HO} = \sqrt{-\frac{a_0}{a_1}} \qquad (4)$$

где $a_1,\ a_0$ - коэффициенты уравнения расстояния (1).

2 Обзор литературы.

Для разработки методологии тестирования мною были изучены работы P.L.Rosin на данную тему [5, 6, 10, 11] для ознакомления с уже выполнявшимися тестами с различными методами нахождения расстояния от точки до квадрики. В ходе изучения данных работ я нашел, что P.L.Rosin использовал следующие наборы данных для тестирования:

1) В работах [5, 10], P.L.Rosin использует только эллипс с полуосями 400 и 100, заданный каноническим уравнением. Рассматривается первый квадрант. В данных работах вычисляется расстояние по известным приближенным формулам в квадранте 1000 x 300 с величиной шага равной 1.

2) В работе [6] P.L.Rosin генерирует 4 набора данных, взяв 38 точек с дуги эллипса (с полуосями 333 и 250, угол сектора 200 градусов) и наложив на них нормально распределенный шум (gaussian noise) в 4 различных направлениях.

3) В работе [11] он сгенерировал 32000 наборов данных для различных эллипсов, в каждом из которых от 10 до 26 точек. Рассмотрены эллипсы с малой полуосью $b = 100$, большой полуосью $a \in [200 ; 450]$; все точки сгенерированы в секторе эллипса, стянутом углом $\theta \in [1 ; 2.5]$ радиан. На данные наложен нормально распределенный шум с дисперсией $\sigma \in [0 ; 40]$.

Используемые P.L.Rosin наборы данных не в полной мере подходят для целей нашего тестирования, так как ёще на ранних стадиях исследования формулы (3) в двумерном случае было установлено, что качество приближения расстояния имеет сильную зависимость от эксцентриситета эллипса, особенно при нахождении расстояния от точки вблизи вершины эллипса, находящейся на большей оси. В связи с вышеуказанными фактами, генерировать наборы данных методами, подобными 1) и 2), является нецелесообразным, так как качество вычисления расстояния для одного конкретного эллипса не дает достаточного количества информации для какого-либо анализа применимости исследуемой формулы.

Наиболее приемлемым для нашего тестирования представлялся способ генерирования данных, подобный 3) варианту, однако необходимо увеличить количество точек в наборах данных. Кроме того, необходимо обобщить методы, рассмотренные выше, на случай трехмерного пространства.

3 Основная часть.

3.1 Тестирование в R^2.

На данном этапе тестирование и сравнение формул проведено для эллипсов, уравнения которых записаны в канонической форме. На каждом рассмотренном эллипсе были найдены точки с фиксированным шагом по оси x (шаг выбран равным 0,01), после чего на полученные данные был наложен нормально распределенный шум N(0, 1) и N(0; 1,5) в направлении нормали. После этого от полученных точек нашли приближенное расстояние до эллипса по формулам d_s, d_{HO}, d_4 и точное расстояние.

Распишем более подробно метод нахождение точного расстояния.

Рассмотрим для полиномов

$$f(x) = a_0x^n + a_1x^{n-1} + \ ... \ + a_n,$$

$$g(x) = b_0x^m + b_1x^{m-1} + \ ... \ + b_m,$$

$$a_0 \neq 0, b_0 \neq 0,$$

квадратную матрицу M размерности $n + m$

$$M = \begin{pmatrix} a_0 & a_1 & a_2 & ... & ... & a_n & 0 & ... & 0 & 0 \\ 0 & a_0 & a_1 & ... & ... & a_{n-1} & a_n & ... & 0 & 0 \\ \vdots & & \ddots & & & & & \ddots & & \vdots \\ 0 & 0 & \cdots & a_0 & ... & \cdots & & \cdots & a_{n-1} & a_n \\ 0 & 0 & \cdots & & b_0 & b_1 & \cdots & \cdots & b_{m-1} & b_m \\ 0 & 0 & \cdots & b_0 & b_1 & ... & & \cdots & b_m & 0 \\ \vdots & & & & \cdots & & & & & \vdots \\ 0 & b_0 & \cdots & & & b_m & \cdots & & \cdots & 0 \\ b_0 & \cdots & \cdots & & b_m & 0 & \cdots & & \cdots & 0 \end{pmatrix},$$

следующее выражение

$$\mathcal{R}(f, g) = (-1)^{\frac{n(n-1)}{2}} \det M$$

называется результантом (в форме Сильвестра).

Тогда выражение

$$\mathcal{D}(f) = \frac{(-1)^{\frac{n(n-1)}{2}}}{a_0}\ \mathcal{R}(f, f')$$

называется дискриминантом полинома f.

Пусть дано уравнение квадрики

$$\mathrm{G(X)} = \mathrm{X^T AX} + 2\mathrm{B^T X} - 1 = 0.$$

Тогда квадрат расстояния от неё до некоторой точки X_0 не принадлежащей ей ($\mathrm{G}(X_0) \neq 0$) равен наименьшему положительному корню уравнения расстояния

$$\mathcal{F}(z) = 0, \text{при } \mathcal{F}(z) = \mathcal{D}_\mu\big(\Phi(\mu, z)\big),$$

где

$$\Phi(\mu, z) = \det\left(\begin{bmatrix} A & B \\ B^T & -1 \end{bmatrix} + \mu \begin{bmatrix} -E & X_0 \\ X_0 & z - X_0^T X_0 \end{bmatrix}\right).$$

Здесь $\mathcal{D}_\mu\big(\Phi(\mu, z)\big)$ - дискриминант полинома $\Phi(\mu, z)$ относительно переменной μ.

В связи с предположением о том, что точность приближения расстояния зависит от значения эксцентриситета эллипса, полученные данные были разделены на несколько групп, что можно увидеть в таблицах 1–4.

Таблица 1. Среднее значение относительной ошибки при N(0,1)

эксцентриситет	d_{HO}	d_s	d_4
0 – 0,5041	0,03%	1,98%	0,08%
0,5041 – 0,6000	0,07%	1,97%	0,1%
0,6000 – 0,9035	0,29%	2,14%	0,12%
0,9035 – 0,9712	1,16%	4,1%	0,57%

Таблица 2. Максимальное значение относительной ошибки при N(0,1)

эксцентриситет	d_{HO}	d_s	d_4
0 – 0,5041	0,87%	13,13%	2,08%
0,5041 – 0,6000	2,01%	115%	115%
0,6000 – 0,9035	347,8%	101,2%	100,1%
0,9035 – 0,9712	145,5%	29,49%	94,26%

Таблица 3. Среднее значение относительной ошибки при N(0; 1,5)

эксцентриситет	d_{HO}	d_s	d_4
0 – 0,5041	0,07%	2,99%	0,17%
0,5041 – 0,6000	0,17%	2,94%	0,16%
0,6000 – 0,9035	0,68%	3,21%	0,27%
0,9035 – 0,9712	2,23%	5,95%	1,43%

Таблица 4. Максимальное значение относительной ошибки при N(0; 1,5)

эксцентриситет	d_{HO}	d_s	d_4
0 – 0,5041	2,63%	20,74%	25,79%
0,5041 – 0,6000	6,02%	31,27%	31,27%
0,6000 – 0,9035	402%	100%	100%
0,9035 – 0,9712	1129%	51,7%	96,6%

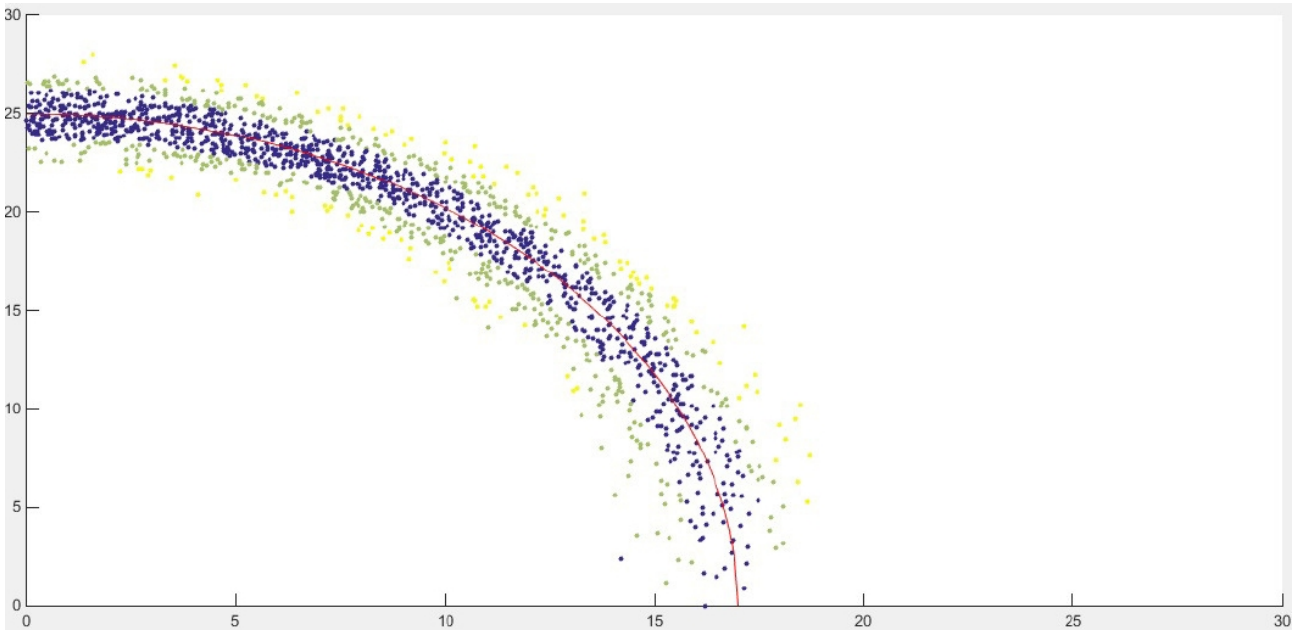


Рис.1. Пример тестовых данных при N(0,1)

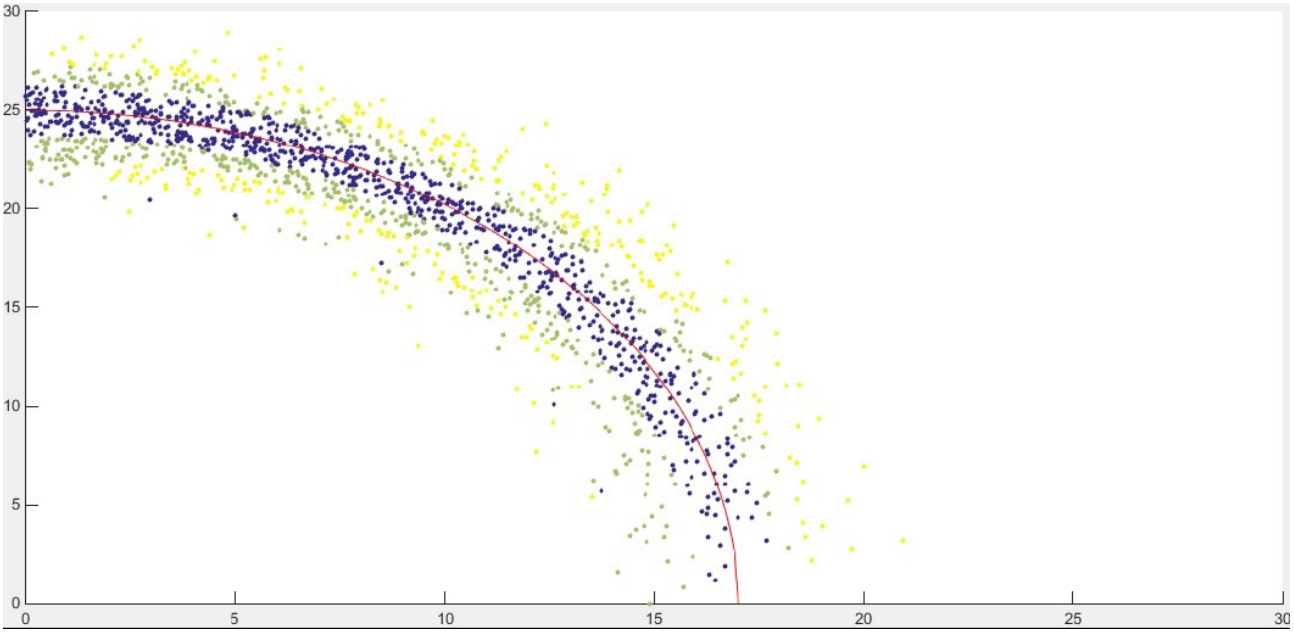


Рис.2. Пример тестовых данных при N(0;1,5)

На рисунках 1 и 2 точки обозначены тремя разными цветами в зависимости от ошибки вычисления расстояния по формуле d_4: синие - ошибка менее 0,1%, зеленые - от 0,1% до 0,3%, желтые -более 0,3%.

3.2 Тестирование в R^3.

Исследование формул, описанных во втором разделе данной работы, производилось в трехмерном пространстве для эллипсоидов, уравнения которых записаны в канонической форме

$$X^T AX - 1 = 0,$$

$$X = (x_1, x_2, x_3)^T,$$

$$A = \begin{pmatrix} \frac{1}{a^2} & 0 & 0 \\ 0 & \frac{1}{b^2} & 0 \\ 0 & 0 & \frac{1}{c^2} \end{pmatrix},$$

или

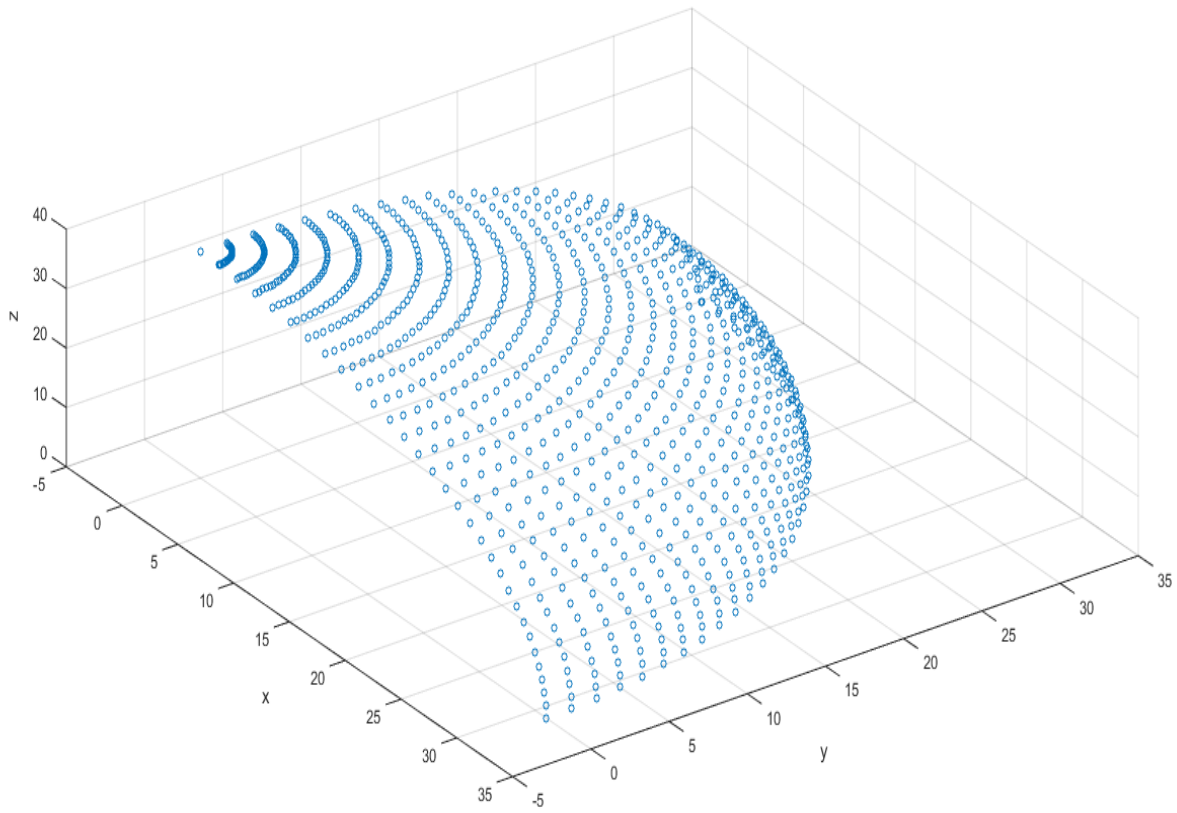
$$\frac{{x_1}^2}{a^2} + \frac{{x_2}^2}{b^2} + \frac{{x_3}^2}{c^2} = 1.$$

Для нахождения точек, принадлежащих эллипсоиду, введем параметризацию

$$\begin{cases} x_1 = a\cos u \cos v, \\ x_2 = b\cos u \sin v, \\ x_3 = c\sin u, \end{cases}$$

$$u \in \left[-\frac{\pi}{2}; \frac{\pi}{2}\right], v \in [0;\ 2\pi)$$

Для каждого эллипсоида варьированием параметров u и v находился набор точек, принадлежащих данному эллипсоиду(~1000 точек для каждого). После этого в каждой из полученных точек была найдена нормаль к эллипсоиду, а на ней точки, смещенные на заданное расстояние (выбиралось смещение равное -1, -2, -3, 1, 2, 3; здесь отрицательное смещение означает, что точка находится на заданном расстоянии внутри поверхности, положительное - снаружи). Затем от полученных точек вычислялось расстояние до эллипсоида по формулам (2) - (4). В исследовании в силу симметрии для каждого эллипсоида рассматривался только один октант ($u \in \left[0; \frac{\pi}{2}\right], v \in [0; \frac{\pi}{2}]$).



На рисунке выше представлен пример полученных тестовых данных для эллипсоида с полуосями 30, 31, 37.

3.3 О классификации эллипсоидов.

В разделе, посвященном исследованию формул в двумерном случае, была замечена зависимость между качеством аппроксимации расстояния и эксцентриситетом эллипса. Эксцентриситет - числовая характеристика кривой второго порядка, которая в случае эллипса вычисляется следующим образом

$$e = \sqrt{1 - \frac{a^2}{b^2}},$$

где a и b - малая и большая полуоси эллипса соответственно.

Однако в трехмерном случае у данной характеристики нет прямого аналога, поэтому я предлагаю использовать в качестве "альтернативы" в трехмерном случае следующее выражение

$$E = \sqrt{1 - \frac{(a^2 + b^2)}{2c^2}} \qquad (5),$$

где a, b и c - малая, средняя и большая полуоси эллипсоида соответственно. Данное выражение имеет ряд свойств, аналогичных свойствам эксцентриситета для эллипса:

1) $E \in [0; 1)$

2) Если $a = b = c$, то E = 0.

Данная величина использовалась для разбиения полученных тестовых данных на различные подгруппы.

3.4 Результаты тестирования в R^3.

Ниже представлены графики, построенные по полученным в ходе тестирования данным. По оси абсцисс указаны значения величины E, по оси ординат - относительной ошибки в процентах. На данных графиках зеленой линией обозначены значения, полученные по формуле d_{HO}, синей - d_s, красной - d_4.

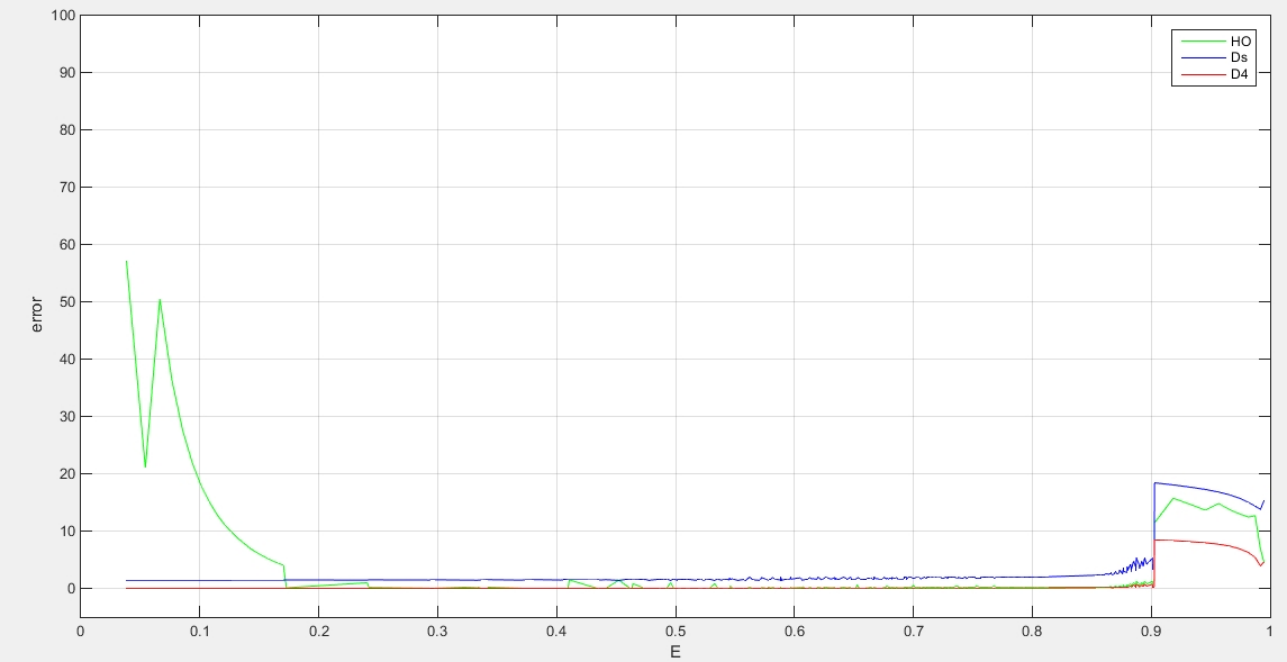


Рис.3. Среднее значение ошибки для расстояния +1.

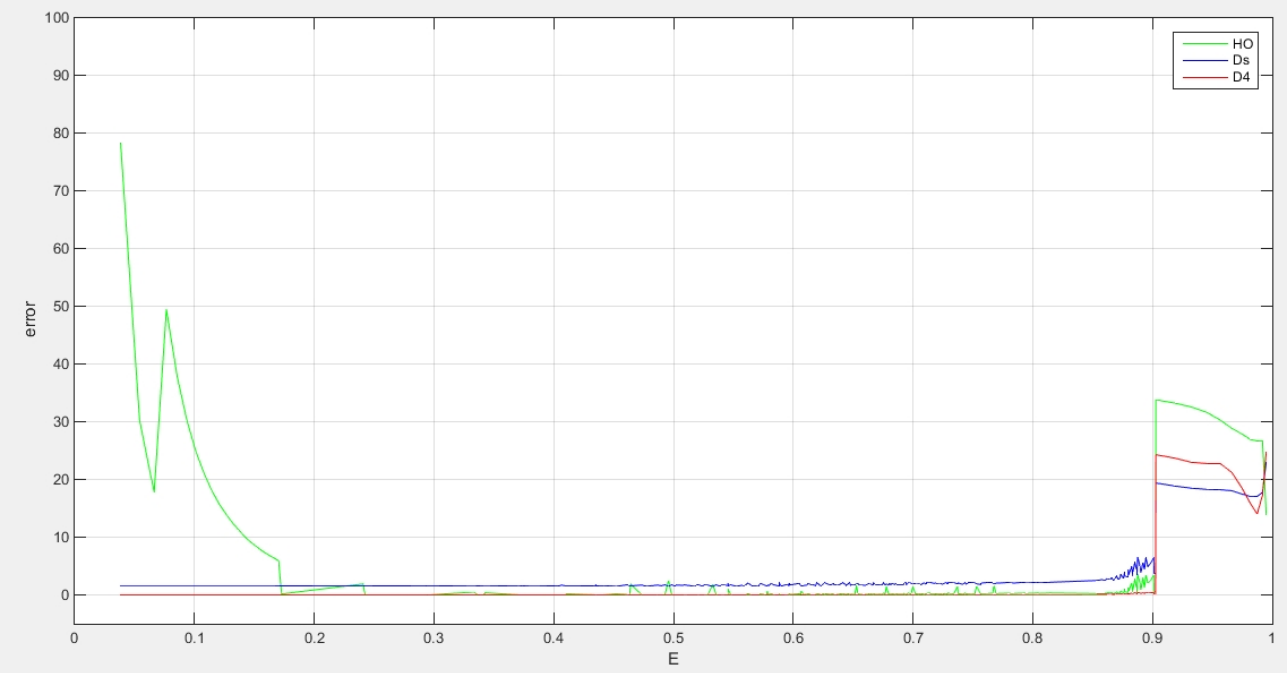


Рис.4. Среднее значение ошибки для расстояния -1.

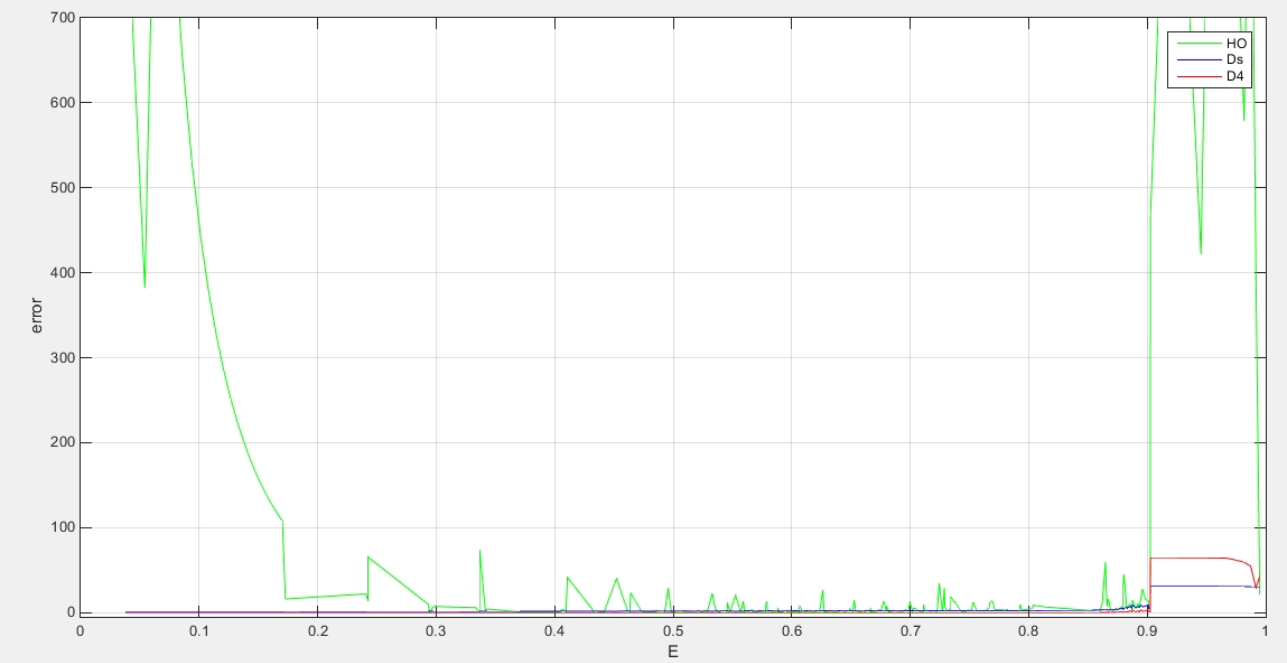


Рис.5. Максимальное значение ошибки для расстояния +1.

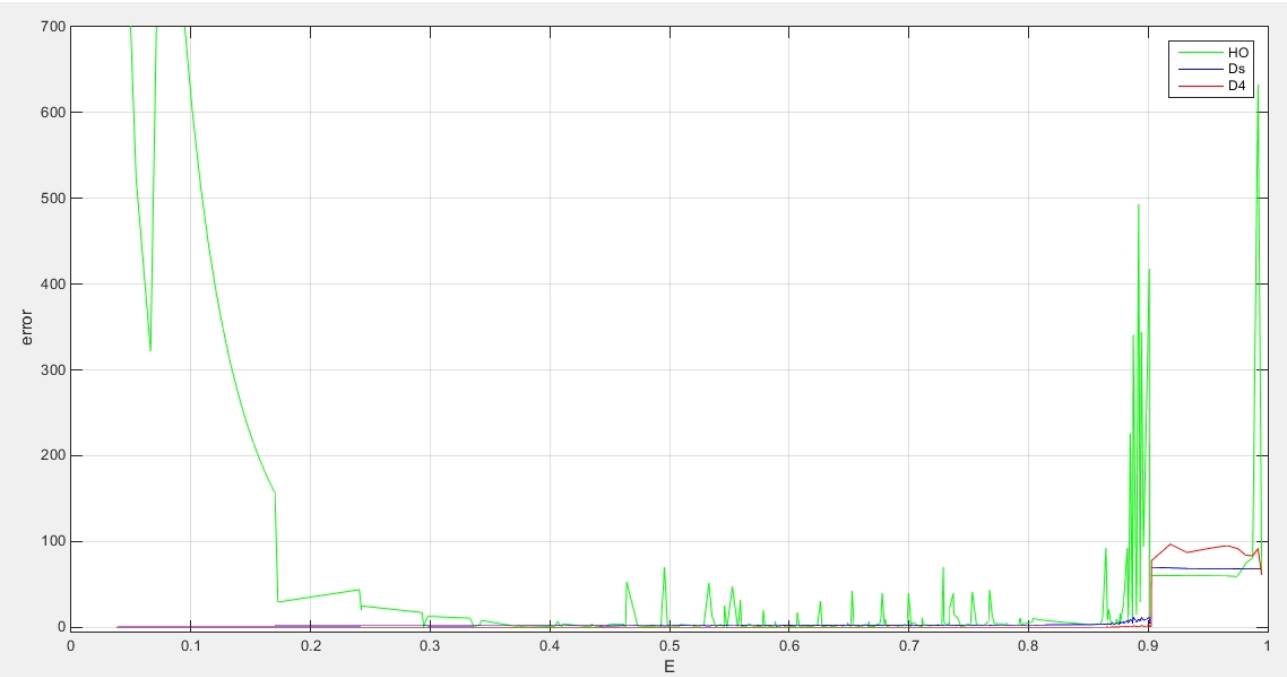


Рис.6. Максимальное значение ошибки для расстояния -1.

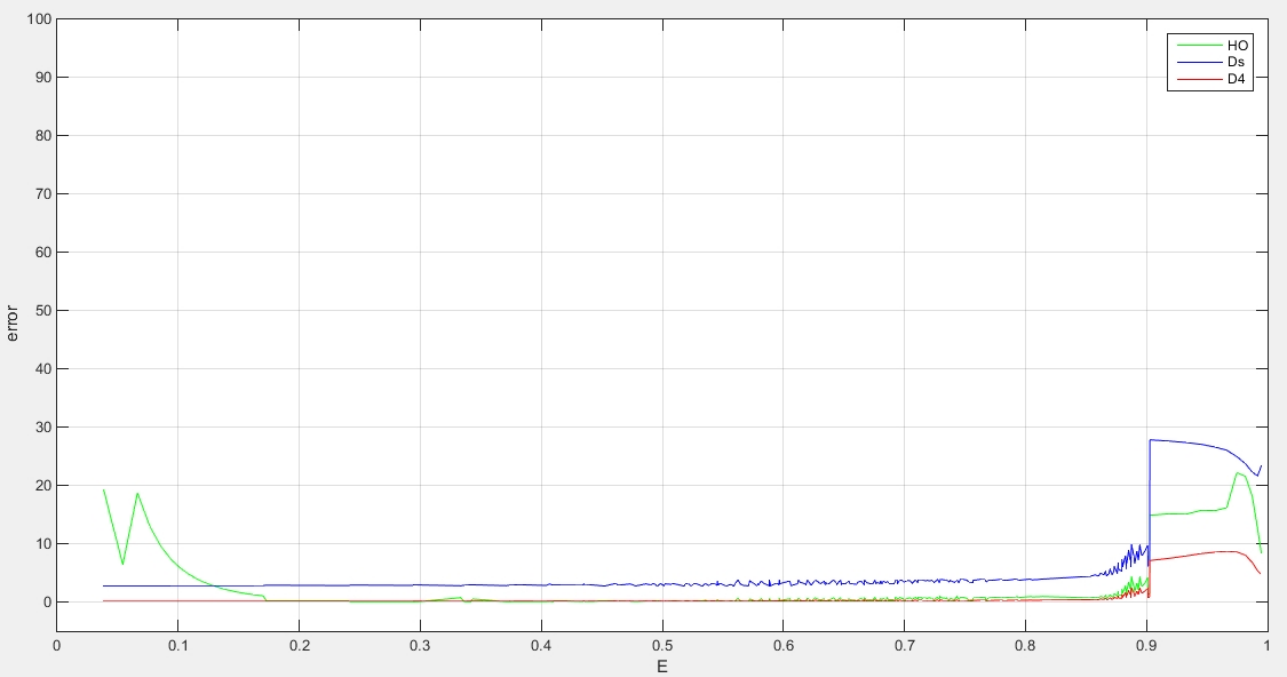


Рис.7. Среднее значение ошибки для расстояния +2.

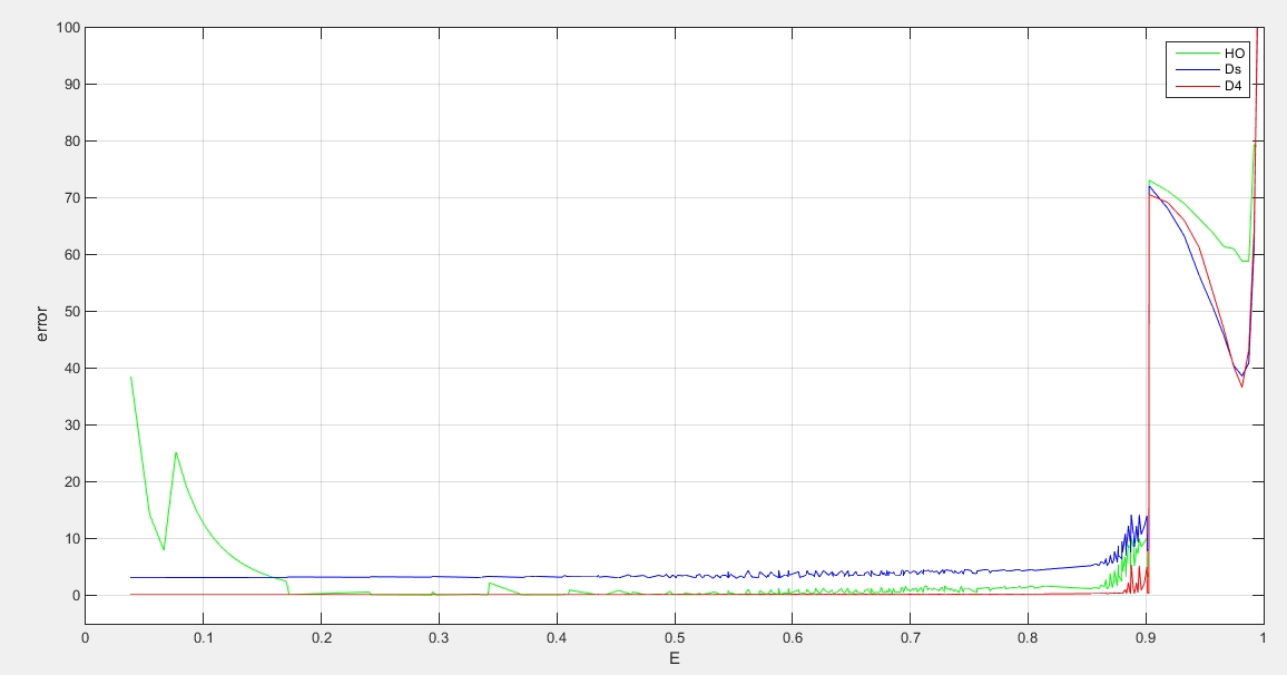


Рис.8. Среднее значение ошибки для расстояния -2.

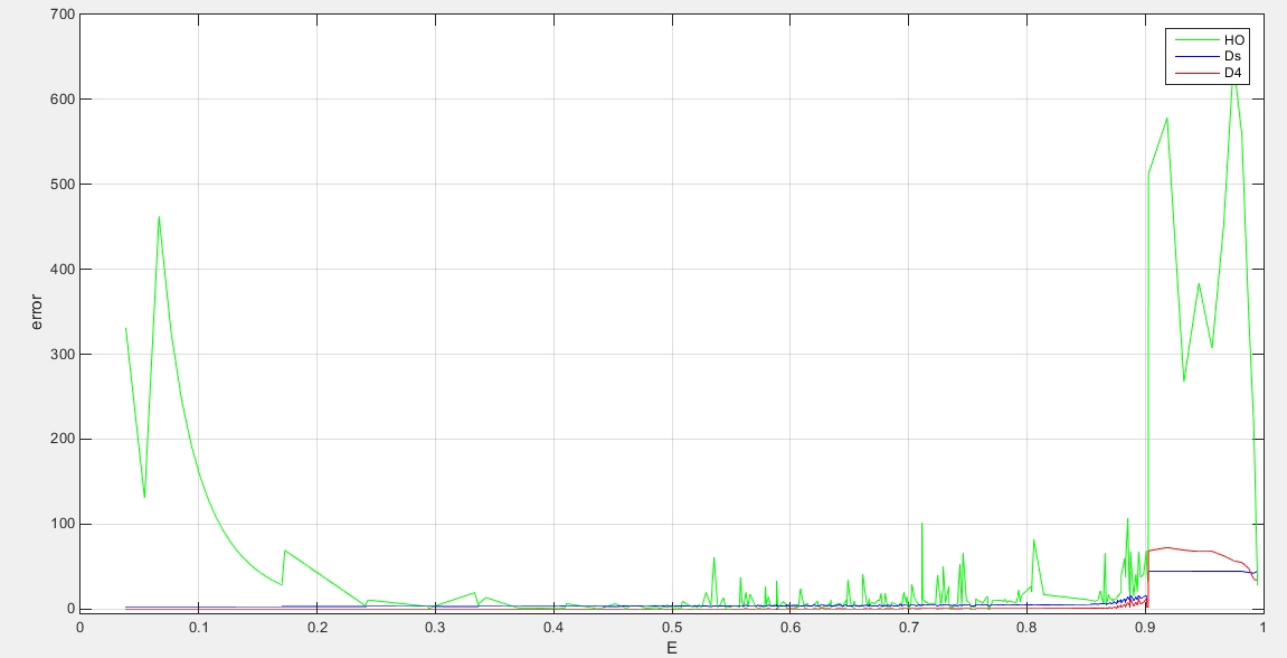


Рис.9. Максимальное значение ошибки для расстояния +2.

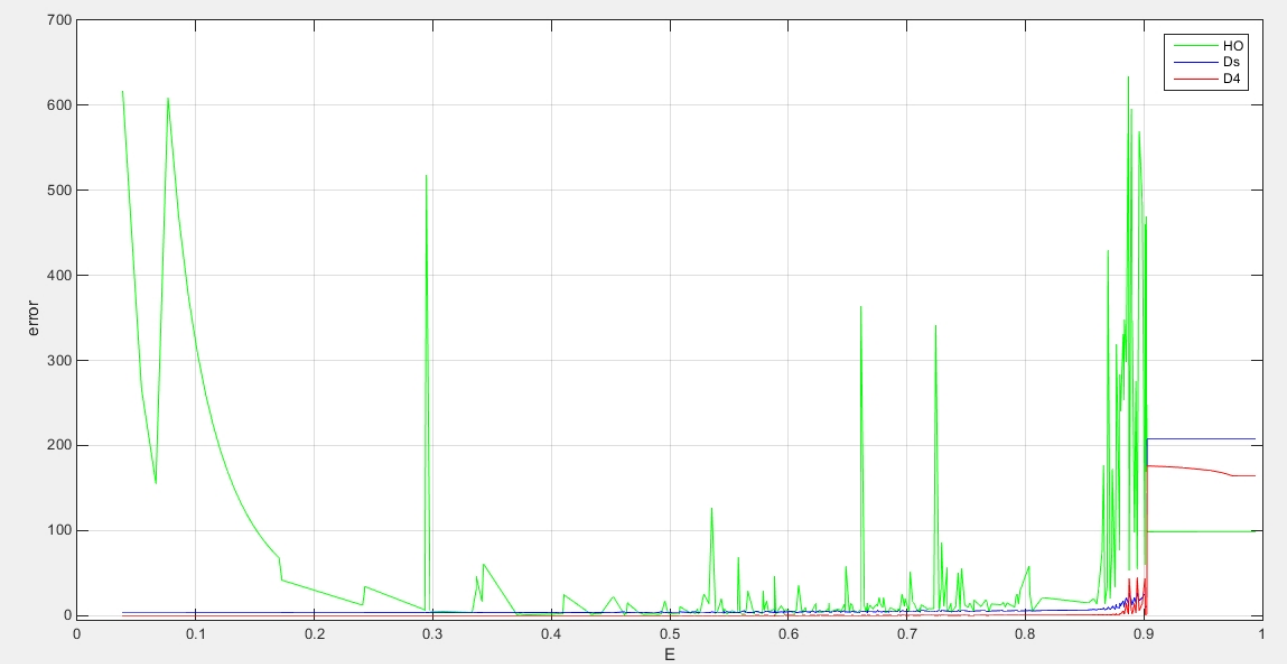


Рис.10. Максимальное значение ошибки для расстояния -2.

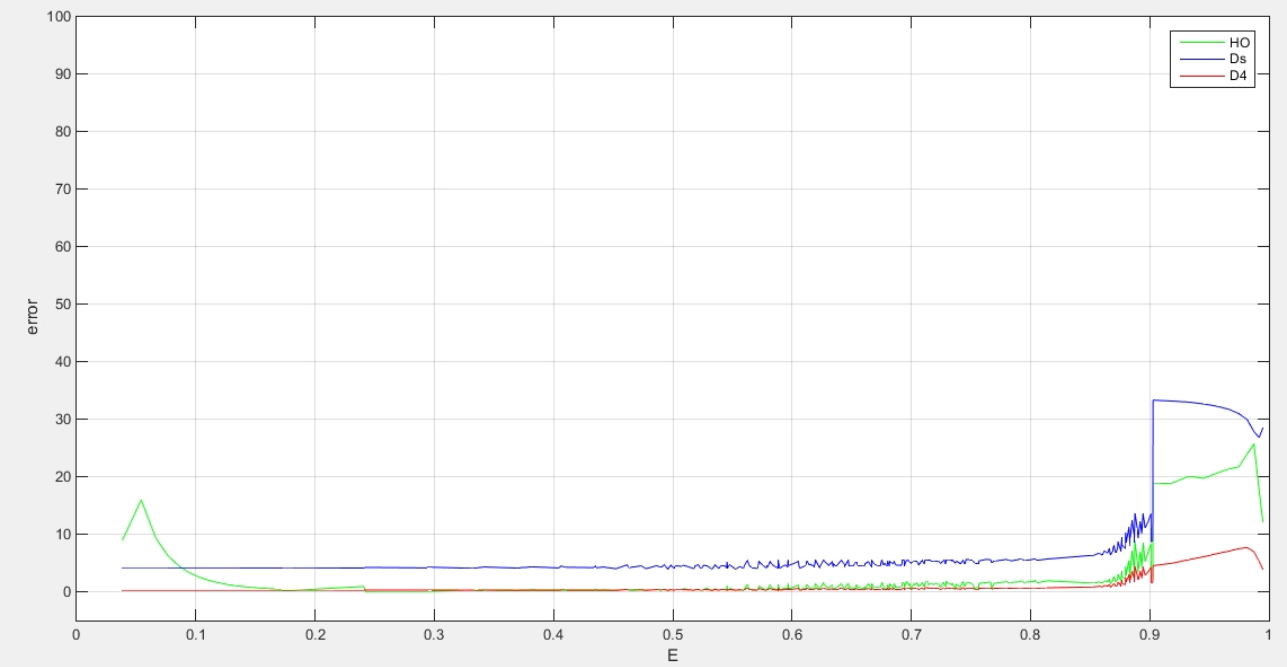


Рис.11. Среднее значение ошибки для расстояния +3.

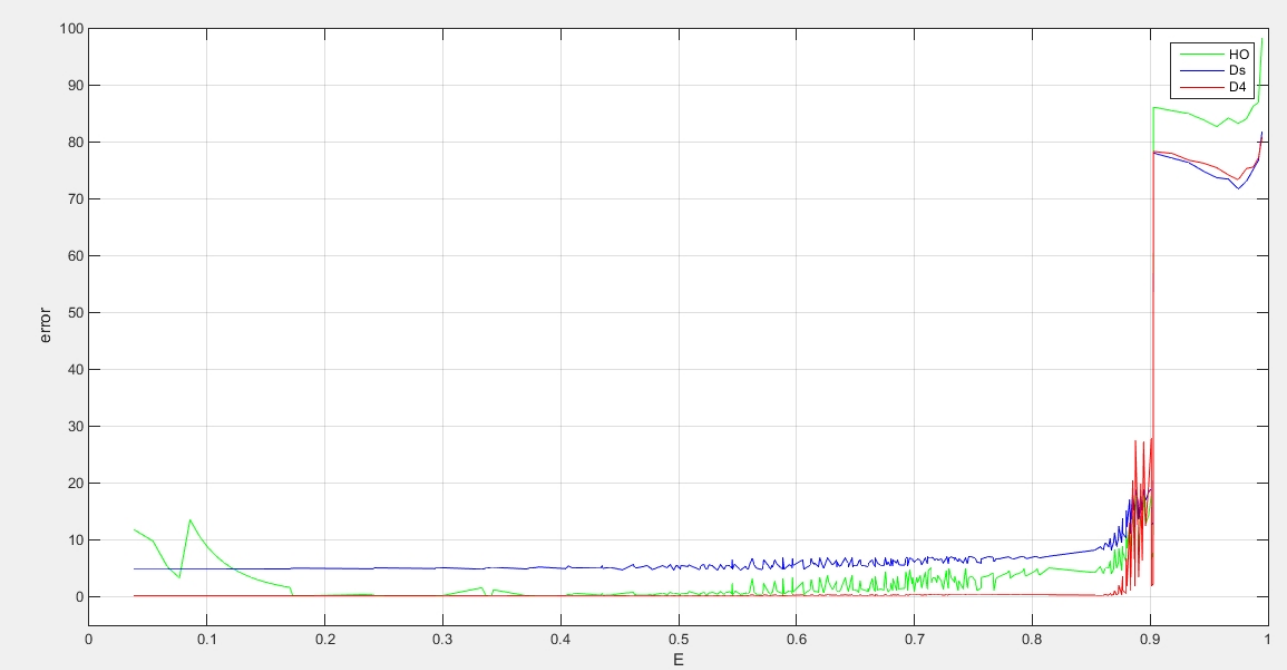


Рис.12. Среднее значение ошибки для расстояния -3.

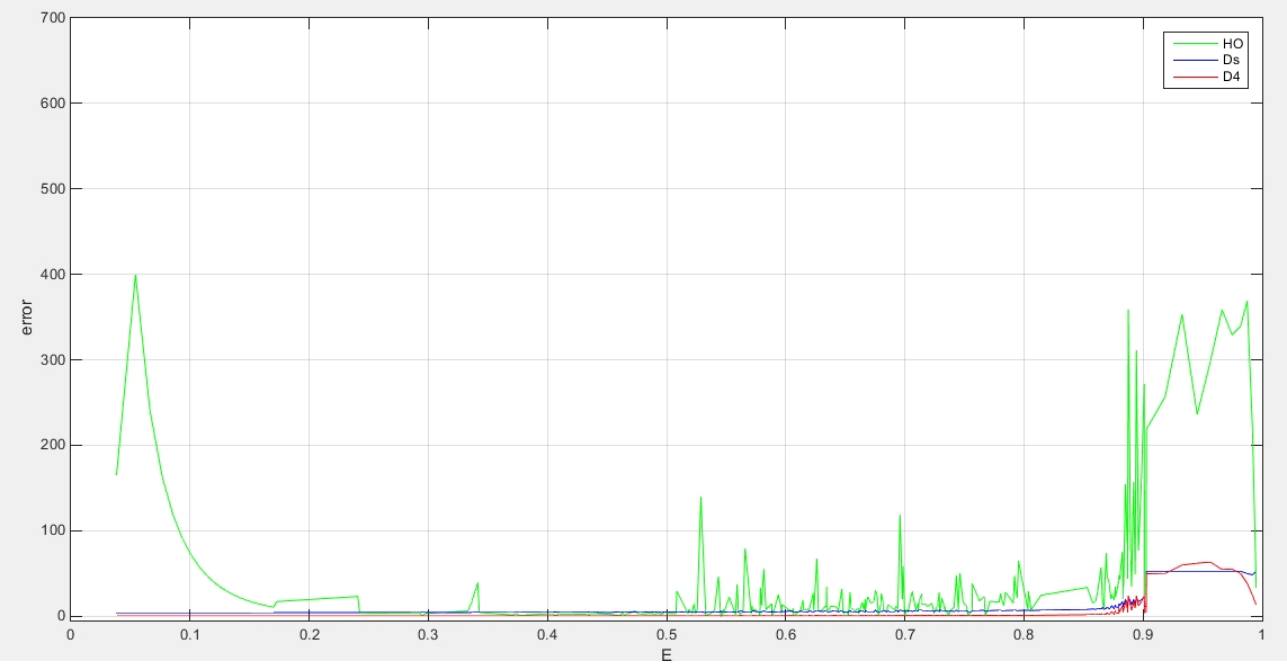


Рис.13. Максимальное значение ошибки для расстояния +3.

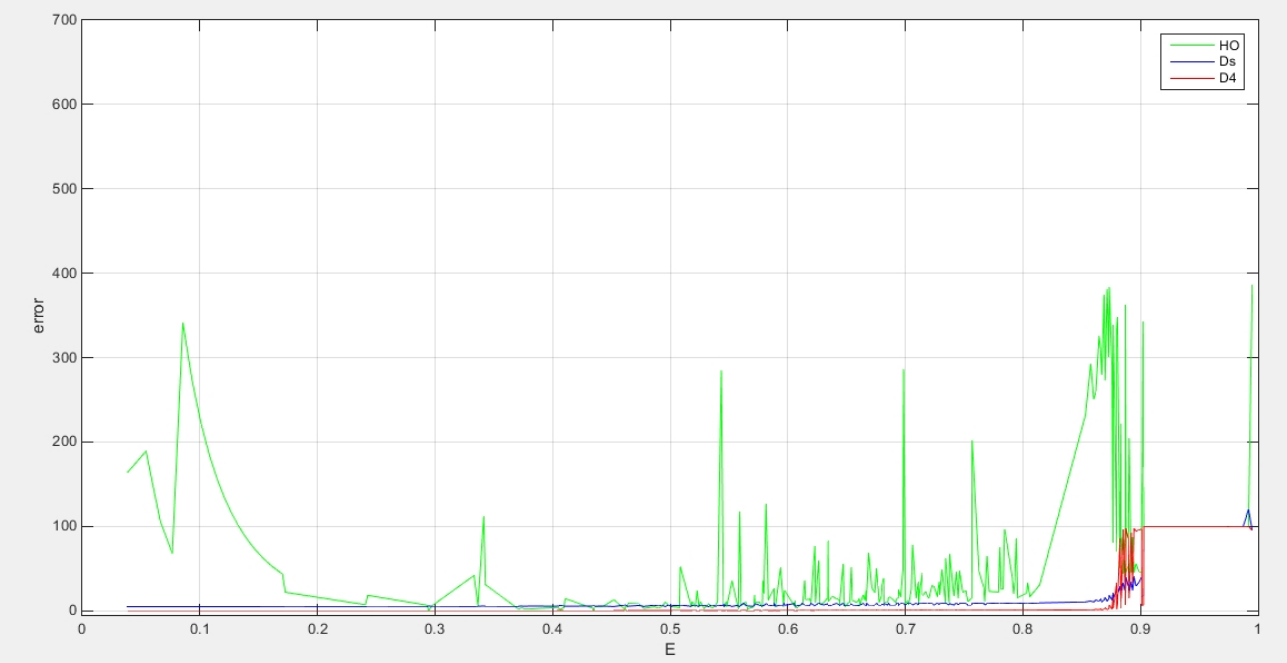


Рис.14. Максимальное значение ошибки для расстояния -3.

Таблица 5. Выборочное среднее относительной ошибки.

расстояние	E	d_{HO}	d_s	d_4
-1	0 - 0,1	40,6572	1,5190	0,0317
	0,1 - 0,2	11,4174	1,5338	0,0323
	0,2 - 0,3	0,3514	1,5819	0,0344
	0,3 - 0,4	0,1112	1,5832	0,0345
	0,4 - 0,5	0,2507	1,6291	0,0373
	0,5 - 0,6	0,1415	1,6940	0,0421
	0 ,6 - 0,7	0,2145	1,8295	0,0515
	0,7 - 0,8	0,3230	2,0493	0,0662
	0,8 - 0,9	1,0240	3,6752	0,2042
	0,9 - 1	19,8772	14,0806	14,3468
1	0 - 0,1	35,7313	1,4321	0,0325
	0,1 - 0,2	8,0381	1,4452	0,0331
	0,2 - 0,3	0,2092	1,4879	0,0353
	0,3 - 0,4	0,0751	1,4891	0,0354
	0,4 - 0,5	0,2493	1,5302	0,0384
	0,5 - 0,6	0,0940	1,5891	0,0438
	0 ,6 - 0,7	0,1399	1,7112	0,0540
	0,7 - 0,8	0,2107	1,9065	0,0703
	0,8 - 0,9	0,4930	3,2366	0,2613
	0,9 - 1	8,5203	12,3377	4,7982
-2	0 - 0,1	20,0405	3,1333	0,1177
	0,1 - 0,2	5,3104	3,1648	0,1197
	0,2 - 0,3	0,2546	3,2672	0,1265
	0,3 - 0,4	0,3365	3,2693	0,1269
	0,4 - 0,5	0,2905	3,3661	0,1361
	0,5 - 0,6	0,3783	3,5035	0,1513
	0 ,6 - 0,7	0,6880	3,7892	0,1809
	0,7 - 0,8	1,1541	4,2567	0,2251
	0,8 - 0,9	4,4840	7,8677	0,8246
	0,9 - 1	47,9436	43,1611	41,3105
2	0 - 0,1	12,3951	2,7844	0,1270
	0,1 - 0,2	2,4236	2,8092	0,1294
	0,2 - 0,3	0,0740	2,8899	0,1375
	0,3 - 0,4	0,1879	2,8916	0,1380
	0,4 - 0,5	0,1530	2,9703	0,1497
	0,5 - 0,6	0,2471	3,0826	0,1705
	0 ,6 - 0,7	0,4221	3,3150	0,2105
	0,7 - 0,8	0,6724	3,6838	0,2728
	0,8 - 0,9	1,6933	6,1047	0,9383

	0,9 - 1	11,5651	19,5763	5,4381
-3	0 - 0,1	9,0651	4,8518	0,2250
	0,1 - 0,2	3,6206	4,9023	0,2279
	0,2 - 0,3	0,1762	5,0660	0,2371
	0,3 - 0,4	0,4714	5,0687	0,2377
	0,4 - 0,5	0,4795	5,2229	0,2496
	0,5 - 0,6	0,9435	5,4404	0,2653
	0 ,6 - 0,7	1,8035	5,8929	0,2971
	0,7 - 0,8	3,2204	6,6400	0,3292
	0,8 - 0,9	8,5303	12,2404	5,4902
	0,9 - 1	62,1070	56,4937	54,9167
3	0 - 0,1	8,0739	4,0634	0,2752
	0,1 - 0,2	1,0473	4,0987	0,2802
	0,2 - 0,3	0,2171	4,2129	0,2972
	0,3 - 0,4	0,1686	4,2157	0,2984
	0,4 - 0,5	0,2614	4,3282	0,3232
	0,5 - 0,6	0,5055	4,4887	0,3676
	0 ,6 - 0,7	0,8674	4,8208	0,4528
	0,7 - 0,8	1,3711	5,3441	0,5838
	0,8 - 0,9	3,4437	8,6638	1,8263
	0,9 - 1	14,9576	24,3206	4,7850

Таблица 6. Максимальное значение относительной ошибки.

расстояние	E	d_{HO}	d_s	d_4
-1	0 - 0,1	1140,7783	1,5290	0,0321
	0,1 - 0,2	601,4052	1,6124	0,0355
	0,2 - 0,3	43,8024	1,7239	0,0403
	0,3 - 0,4	10,7193	1,7241	0,0403
	0,4 - 0,5	69,328	2,0833	0,0578
	0,5 - 0,6	51,2684	2,6818	0,1051
	0 ,6 - 0,7	42,2266	2,7672	0,1198
	0,7 - 0,8	69,6213	2,7927	0,1238
	0,8 - 0,9	492,5587	11,5092	2,0261
	0,9 - 1	632,3876	69,8439	96,7988
1	0 - 0,1	1159,8297	1,4409	0,0329
	0,1 - 0,2	443,1732	1,5147	0,0365
	0,2 - 0,3	65,4274	1,6127	0,0414
	0,3 - 0,4	74,3260	1,6129	0,0414
	0,4 - 0,5	41,4709	1,9231	0,0601
	0,5 - 0,6	23,0695	2,4386	0,1129
	0 ,6 - 0,7	26,0918	2,5211	0,1300
	0,7 - 0,8	34,4460	2,5447	0,1347
	0,8 - 0,9	58,9326	9,0820	3,0941
	0,9 - 1	1293,707	30,97048	64,6537
-2	0 - 0,1	616,4342	3,1545	0,1190
	0,1 - 0,2	310,2482	3,3328	0,1303
	0,2 - 0,3	517,9350	3,5714	0,1457
	0,3 - 0,4	60,6042	3,5714	0,1457
	0,4 - 0,5	24,2845	4,3478	0,1999
	0,5 - 0,6	126,0508	5,6475	0,3462
	0 ,6 - 0,7	363,5602	5,8192	0,4003
	0,7 - 0,8	340,3043	5,8722	0,4149
	0,8 - 0,9	633,9819	26,1009	44,0216
	0,9 - 1	469,5222	207,5133	175,9498
2	0 - 0,1	461,9825	2,8011	0,1286
	0,1 - 0,2	155,5689	2,9410	0,1421
	0,2 - 0,3	10,6226	3,1246	0,1609
	0,3 - 0,4	19,2282	3,1250	0,1609
	0,4 - 0,5	6,2810	3,7037	0,2315

	0,5 - 0,6	59,9756	4,6670	0,4359
	0 ,6 - 0,7	41,0528	4,8265	0,5058
	0,7 - 0,8	101,8169	4,8750	0,5255
	0,8 - 0,9	105,8854	16,3158	11,8448
	0,9 - 1	643,8871	44,6267	72,3677
-3	0 - 0,1	341,0503	4,8859	0,2269
	0,1 - 0,2	221,1002	5,1716	0,2418
	0,2 - 0,3	18,4046	5,5546	0,2589
	0,3 - 0,4	111,5323	5,5556	0,2592
	0,4 - 0,5	14,3258	6,8182	0,3266
	0,5 - 0,6	284,9432	8,9449	0,5494
	0 ,6 - 0,7	286,7258	9,2052	0,6786
	0,7 - 0,8	201,9236	9,2854	0,7113
	0,8 - 0,9	383,4197	40,0828	97,7414
	0,9 - 1	386,3787	120,2041	99,9953
3	0 - 0,1	399,4853	4,0871	0,2785
	0,1 - 0,2	70,8400	4,2854	0,3067
	0,2 - 0,3	23,0513	4,5448	0,3455
	0,3 - 0,4	38,9161	4,5455	0,3456
	0,4 - 0,5	6,0337	5,3571	0,4921
	0,5 - 0,6	138,7409	6,7102	0,9253
	0 ,6 - 0,7	118,3922	6,9472	1,0827
	0,7 - 0,8	63,7752	7,0145	1,1283
	0,8 - 0,9	358,7660	22,1667	23,7343
	0,9 - 1	368,6365	52,5844	63,2720

4 Выводы и заключение.

Анализируя данные, полученные при проведении тестирования, можно заключить, что среднее значение относительной ошибки для метода (3) в большинстве случаев оказывается меньше, чем для конкурирующих методов (2) и (4). Можно отметить довольно странную закономерность: качество приближения расстояния методом (4) немного улучшается при увеличении расстояния от 1 до 3.

Несмотря на то, что приближение расстояния методом (4) в среднем имеет точность, довольно близкую к методу (3), величина максимальной ошибки (4) превосходит в разы аналогичные значения для метода (3).

В ходе выполнения тестирования выявлено, что в некоторых случаях коэффициенты уравнения расстояния (1) могут в определенных областях обращаться в 0, однако небольшие отклонения точек от данных областей позволяют получить не нулевые значения, но качество получаемого методом (4) приближения расстояния для них являются крайне неудовлетворительными, что свидетельствует о невозможности применения метода (4). В таких случаях для прикладных задач лучше использовать другие формулы нахождения приближенного расстояния.

Также важно отметить, что максимальное значение относительной ошибки метода (3) в большинстве случаев намного меньше, чем для конкурирующих методов, однако эта величина начинает значительно возрастать при $E > 0{,}85$, в следствие чего в большинстве рассматриваемых случаев превосходит максимальную относительную ошибку метода (2) (исходя из выражения для величины E, это относится к эллипсоидам, у которых одна из полуосей которых *намного* больше двух других).

Стоит заметить, что качество аппроксимации расстояния для внутренних точек эллипсоидов несколько хуже, чем для внешних для всех рассмотренных методов.

Несмотря на то что методы (3) и (4) в среднем дают более качественную оценку расстояния, чем метод (2), у них есть серьезный недостаток по сравнению с методом (2) - оценка расстояния для (3) и (4) определена не в каждой точке пространства (в обоих случаях подкоренное значение может оказаться отрицательным; попытки взять подкоренное значение по модулю не приводят к сколько-нибудь значимым результатам), однако для метода (4) данные точки начинают появляться только при довольно большом значении E.

Для иллюстрации данного факта ниже приведен график зависимости количества таких точек от значения E.

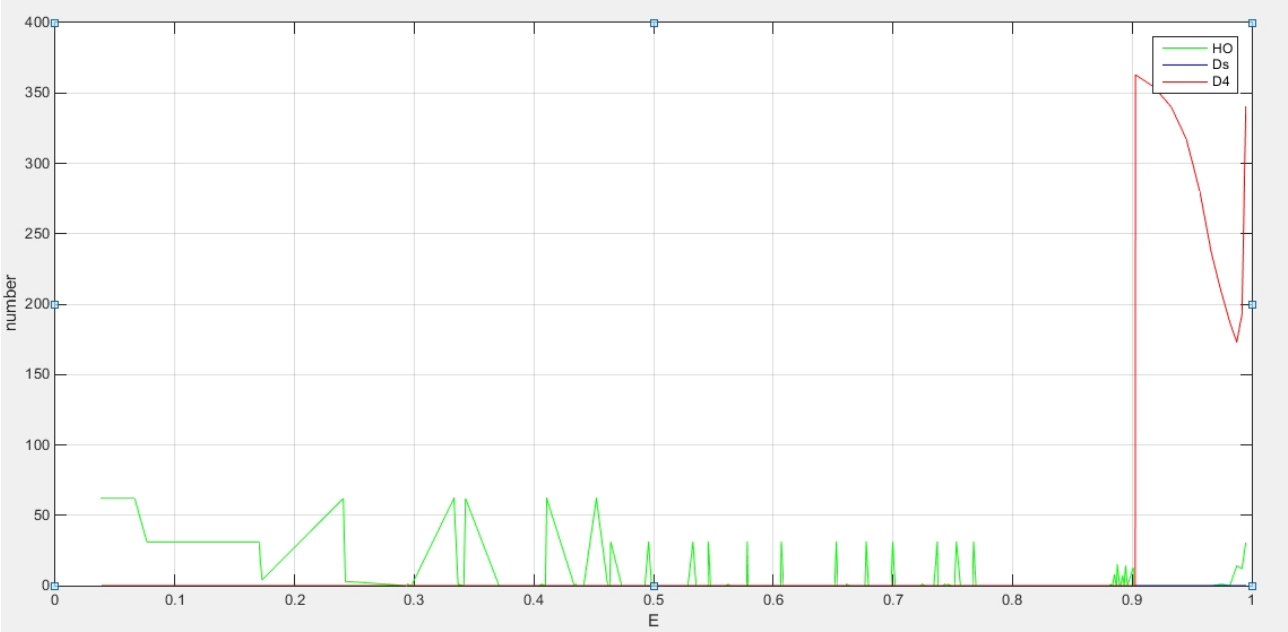


Рис.15. Количество точек, для которых оценка расстояния не определена при расстояние -1.

Таким образом, в средах maple и matlab было разработано программное обеспечение для проведения компьютерного тестирования формулы (3) и её конкурентов (2) и (4). Программное обеспечение позволяет находить приближенное расстояние с помощью методов (2), (3) и (4), генерировать наборы тестовых данных для различных эллипсов/эллипсоидов. Группировать эллипсы по эксцентриситету в R^2 или по величине E (5) в R^3, строить графики для визуализации полученных данных. Проведен анализ тестовых данных, на основе которого показано преимущество метода (3) по сравнению с конкурирующими методами в большинстве рассмотренных случаев (при $E < 0{,}85$). При $E > 0{,}9$ оценка, получаемая методом (3) оказывается неопределенной для довольно большого числа точек, поэтому, в данном случае на практике более целесообразно будет использовать метод (2). Так, все поставленные задачи решены, результаты представлены в данной работе.

Список литературы.

[1] Sun Q. B., Lam C. P., Wu J. K. A practical automatic face recognition system // Face Recognition. From Theory to Applications. Springer, 1998. P. 537–546.

[2] Rosin P. L. Measuring shape: Ellipticity, rectangularity, and triangularit // Machine Vision and Applications. 2003. No 14 (3). P. 172–184.

[3] Heikkila J. Geometric camera calibration using circular control points // IEEE Trans. PAMI. 2000. No 22(10). P. 1066–1077.

[4] Uteshev A. Yu., Yashina M. V. Metric problems for quadrics in multidimensional space // J. Symbolic Computation. 2015. Vol. 68, part 1. P. 287–315.

[5] Rosin P.L. Assessing error of fit functions for ellipses // Graphical Models and Image Processing. 1996. Vol. 58, no. 5, P. 494-502.

[6] Rosin P.L. Ellipse fitting using orthogonal hyperbolae and Stirling's oval // Graphical Models and Image Processing. 1998. Vol. 60, no. 3, P. 209-213.

[7] Sampson P. D. Fitting conic sections to very scattered data. An iterative refinement to the Bookstein algorithm // Computer Vision, Graphics and Image Processing. 1982. No 18. P. 97–108.

[8] Uteshev A. Yu., Goncharova M. V. Point-to-ellipse and point-to-ellipsoid distance equation analysis // J. Comput. Appl. Math. 2018.Vol. 328. P. 232–251.

[9] Harker M., O'Leary P. First order geometric distance (the myth of Sampsonus) // British Machine Vision Conf. 2006. P. 87–96.

[10] Rosin P.L. Further five-point fit ellipse fitting // Graphical Models and Image Processing. 1999. Vol. 61, no. 5, P. 245-259.

[11] Rosin P.L. Evaluating Harker and O′Leary′s distance approximation for ellipse fitting // Pattern Recognition Letters. 2007. Vol. 28, no. 13, P. 1804-1807.

Приложение

createExperimentalData3D.mw

```
with(LinearAlgebra) :
with(linalg) :
with(ExcelTools) :
with(stats) :
Digits = 10 :

#Сначала в виде готовой формулы получим уравнение расстояния(F_tr_dist)

    #в нем координаты точки(x1, y1, z1) и коэффициенты a, b, c в формуле эллипсоида
    (x^2/a^2 + y^2/b^2 + z^2/c^2 -1 = 0 )
# параметры, которые будут подставляться в полученную формулу
    B := ⟨0; 0; 0⟩ :
    A := Matrix([[1/a1^2, 0, 0], [0, 1/b1^2, 0], [0, 0, 1/c1^2]]) :
    x := ⟨x1; y1; z1⟩ :
    help1 := Matrix([[A, B], [Transpose(B), -1]]) :
    help2 := Matrix( [[diag(-1, -1, -1), x], [Transpose(x), z - (Transpose(x)
    .x)[1][1]]]) :
    Fi := Determinant(help1 + q * help2) :
    F_tr_dist := discrim(Fi, q) :
unassign('A','B','Fi','help1','help2','x') :
# Дальше идут 3 функции подсчета приближенного расстояния.

# расстояние по формуле dho(Harkery and O'Leary)
# x0,y0,z0 - координаты точки, a0,b0,c0 - значения полуосей эллипсоида
distanceHO := proc(x0, y0, z0, a0, b0, c0) :
    help := subs(x1 = x0, y1 = y0, z1 = z0, a1 = a0, b1 = b0, c1 = c0, F_tr_dist) :
    if coeff(help, z, 1) ≠ 0 then
        ans := Re(evalf(sqrt(-coeff(help, z, 0)/coeff(help, z, 1)))) :
        if ans = 0 or ans > 15 then
            ans := 'NaN':
        fi:
    else
        ans := 'NaN':
    fi:
    RETURN(ans) :
end:

#расстояние по формуле ds(Sampson's distance)
#Аргумент x0 - вектор (x,y,z), A - матрица квадратичной формы.
distanceD1 :=proc(x0, A) :
    G := x → Transpose(x).A.x - 1 :
    d1 := abs(G(x0)) / (2 * sqrt((Transpose(A.x0).(A.x0)))) :
    RETURN(Re(evalf(d1))) :
end:

#experimental d4
distanceD4 :=proc(x0, A) :
```

```
    G := x → Transpose(x).A.x - 1 :
    d1 := abs(G(x0)) / (2 * sqrt((Transpose(A.x0).(A.x0)))) :
    k := (Transpose(A.x0).A.(A.x0)) :
    grad2 := (Transpose(A.x0).(A.x0)) :
    g := G(x0) :
    d2 := d1 * sqrt( 1 + (g * k) / ( 2 * ( (grad2)^2 )) + ((g^2 * k^2) / (2 * grad2^4))
    + ((5 * k^3 * g^3) / (8 * grad2^6)) ) :
    RETURN(Re(evalf(d2))) :
end:

#функция нахождения точек эллипсоида с помощью параметризации
findEllipsePoint := proc(u, v, a, b, c) :
    uFloat := evalf(u) :
    vFloat := evalf(v) :
    x := a * cos(uFloat) * cos(vFloat) :
    y := b * cos(uFloat) * sin(vFloat) :
    z := c * sin(uFloat) :
    answer := ⟨x, y, z⟩ :
    RETURN(answer) :
end:


# функция нахождения нормали к эллипсоиду в точке x0
getNormalAtPoint := proc(x0, a, b, c) :
    answer := ⟨2 * x0[1] / a^2, 2 * x0[2] / b^2, 2 * x0[3] / c^2⟩ :
    answer := answer / norm(answer, 2) :
    RETURN(answer)
end:

# функция нахождения точки, на расстояние step от эллипсоида в направлении нормали
getStepPoint := proc(myPoint, myNormal, step) :
    answer := myPoint + myNormal * step :
    RETURN(answer) :
end:
```

```
    # функция для нахождения всех точек одного эллипсоида, которые будут
    использоваться в тестировании
evaluateExperimentPointsForEllipse := proc(a, b, c, step, DISTANCE) :
    A := ⟨1/a^2, 0, 0; 0, 1/b^2, 0; 0, 0, 1/c^2⟩ :
    iEnd := Pi/2 :
    jEnd := Pi/2 :
    iStop := iEnd/step :
    jStop := jEnd/step :
    endpoint := convert((iStop + 1) * (jStop + 1), rational) :
    answer := array(1..endpoint, 1..3, [ ]) :
    iterator := 1 :

    for i from 0 to iStop by 1 do
        for j from 0 to jStop by 1 do




            u := i * step :
            v := j * step :
            ellipsePoint := findEllipsePoint(u, v, a, b, c) :
            myNormal := getNormalAtPoint(ellipsePoint, a, b, c) :
            tempPointHolder := getStepPoint(ellipsePoint, myNormal, DISTANCE) :
            answer[iterator, 1] := tempPointHolder[1] :
            answer[iterator, 2] := tempPointHolder[2] :
            answer[iterator, 3] := tempPointHolder[3] :
            iterator := iterator + 1 :
        od:
    od:
    RETURN(Matrix(answer)) :
end:

saveInExcel := proc(data, idName) :
    address := cat(idName, ".xls") :
    Export(Matrix(data), address) :
end:
```

```
doExperiment := proc(first, second, third, WORKPATH, DISTANCE) :
    for a from 1 to Dimension(first) do
        for b from 1 to Dimension(second) do
            for c from 1 to Dimension(third) do
                A := ⟨1/first[a]^2, 0, 0; 0, 1/second[b]^2, 0; 0, 0, 1/third[c]^2⟩ :
                data := evaluateExperimentPointsForEllipse(first[a], second[b], third[c],
    Pi/60, DISTANCE) :
                answer := array(1..Dimension(data)[1], 1..7, [ ]) :
                for i from 1 to Dimension(data)[1] do
                    answer[i, 1] := data[i, 1] :
                    answer[i, 2] := data[i, 2] :
                    answer[i, 3] := data[i, 3] :
                    answer[i, 4] := abs(DISTANCE) :
                    answer[i, 5] := distanceHO(data[i, 1], data[i, 2], data[i, 3], first[a],
    second[b], third[c]) :
                    answer[i, 6] := distanceD1(⟨data[i, 1], data[i, 2], data[i, 3]⟩, A) :
                    answer[i, 7] := distanceD4(⟨data[i, 1], data[i, 2], data[i, 3]⟩, A) :
                od:
                saveInExcel(answer, cat(WORKPATH, "data_", first[a], "_", second[b], "_",
    third[c])) :

            od:
        od:
    od:
end:

doExperimentForAllDistances := proc(first, second, third) :
    for i from -3 to 3 by 1 do
        if i = 0 then
            next:
        fi:




        WORKPATH := cat("C:\\Users\\Влад\\Documents\\Maple\\test_3d\\bez_povorota\\",
    i, "\\") :
        DISTANCE := i :
        doExperiment(first, second, third, WORKPATH, DISTANCE) :
    od:

end:
```

graphics3D.m

```
var_a = [20, 21, 25, 30, 31, 33];
var_b = [25, 26, 27, 31, 32, 33];
var_c = [33, 34, 35, 37, 38, 39];
%------------
var_a2 = [4];
var_b2 = [4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24];
var_c2 = [40];
%------------
var_a3 = [50];
var_b3 = [29, 30, 31];
var_c3 = [10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20];
%------------
var_a4 = [40];
var_b4 = [20];
var_c4 = [13.9, 14, 14.1, 14.2];
%------------
var_a5 = [34];
var_b5 = [34];
var_c5 = [33, 33.05, 33.10, 33.15, 33.20, 33.25, 33.3,↙
33.35, 33.4, 33.45, 33.5, 33.55, 33.6, 33.65, 33.7, 33.75,↙
33.8, 33.85, 33.9, 33.95];
WORKPATH = 'C:↙
\\Users\\Влад\\Documents\\Maple\\test_3d\\bez_povorota\\3\\↙
';

graphic_data_mean = [];
graphic_data_max = [];
for iter=1:5
    if iter == 2
        var_a = var_a2;
        var_b = var_b2;
        var_c = var_c2;
    end
    if iter == 3
        var_a = var_a3;
        var_b = var_b3;
        var_c = var_c3;
    end
```

```
    if iter == 4
        var_a = var_a4;
        var_b = var_b4;
        var_c = var_c4;
    end
     if iter == 5
        var_a = var_a5;
        var_b = var_b5;
        var_c = var_c5;
    end
 for i=1:length(var_a)
    for j=1:length(var_b)
        for k=1:length(var_c)

            exct = getExpirementalExct(var_a(i), var_b(j),↙
var_c(k));

            temp = getData3DForStats([WORKPATH, 'data_',↙
num2str(var_a(i)), '_', num2str(var_b(j)), '_', num2str↙
(var_c(k)),'.xls']);
            graphic_data_mean = [graphic_data_mean; exct,↙

mean(temp)];
            graphic_data_max = [graphic_data_max; exct, max↙
(temp)];
        end
    end
end
end
A_mean = sortrows(graphic_data_mean);
X_mean = A_mean(:,1);
Y1_mean = A_mean(:,2)*100;
Y2_mean = A_mean(:,3)*100;
Y3_mean = A_mean(:,4)*100;
plot(X_mean,Y1_mean,'g',X_mean,Y2_mean,'b',X_mean,↙
```

```
Y3_mean,'r')
axis([0 1 -5 100]);
grid on;
xlabel('E');
ylabel('error');
legend('HO','Ds','D4', 1);
A_max = sortrows(graphic_data_max);
X_max = A_max(:,1);
Y1_max = A_max(:,2)*100;
Y2_max = A_max(:,3)*100;
Y3_max = A_max(:,4)*100;
%plot(X_max,Y1_max,'g',X_max,Y2_max,'b',X_max,Y3_max,'r')
%xlswrite('data\\mean-1.xls',graphic_data_mean);
%xlswrite('data\\max-1.xls',graphic_data_max);
%[Mmean, Mmax] = getRangedStat(graphic_data_mean,↙
graphic_data_max);
%Mmean = Mmean*100;
%Mmax = Mmax *100;
```

```
function [ data ] = getData3DForStats( path )
mydata = xlsread(path);
sorted = [];
iter = 1;
for i=1:length(mydata)
    if (~isnan(mydata(i, 5)) && (mydata(i, 7) ~= 0))
        sorted = [sorted; zeros(1,3)];
        for j=4:6
            sorted(iter, j - 3) = abs(mydata(i, 4) - mydata↙
(i, j + 1))/mydata(i, 4);
        end
        iter = iter + 1;
    end
end
data = sorted;
end
```

```
function [ exct ] = getExpirementalExct(a ,b ,c)
arr = sort([a, b, c]);
small = arr(1);
middle = arr(2);
big = arr(3);
exct = sqrt(1 - (small^2 + middle^2)/(2 * big^2));
end
```

```
function [ data ] = getNaNPoints( path )
mydata = xlsread(path);
sorted = [0 0 0];
for i=1:length(mydata)
    if isnan(mydata(i, 5))
        sorted(1) = sorted(1)+ 1;
    end
    if (mydata(i, 7) == 0)
        sorted(3) = sorted(3)+ 1;
    end
end
data = sorted;
end
```

```
function [ sortedStatMean, sortedStatMax ] = getRangedStat(↙
dataMean, dataMax )
iterator = zeros(1,10);
sortedStatMean = zeros(10,3);
sortedStatMax = zeros(10,3);
sortedDataMean = sortrows(dataMean);
sortedDataMax = sortrows(dataMax);
j = 1;
for i=1:length(sortedDataMean)
    exct = sortedDataMean(i,1);
    close = 0.1*j;
    if exct > close
        iterator(j) = i - 1;
        j = j + 1;

    end
end
iterator(10) = length(sortedDataMean);
```

```
j = 1;
for i=1:length(iterator)
    tempMean = sortedDataMean(j:iterator(i), 2:4);
    tempMax = sortedDataMax(j:iterator(i), 2:4);
    sortedStatMean(i,1:3) = mean(tempMean);
    sortedStatMax(i,1:3) = max(tempMax);
    j = iterator(i) + 1;
end
end
```

createExperimentalData2D.mw

```
with(LinearAlgebra) :
with(linalg) :
with(ExcelTools) :
with(stats) :
Digits = 10 :

#Сначала в виде готовой формулы получим уравнение расстояния(F_tr_dist)

    #в нем координаты точки(x1,y1) и коэффициенты  a и b в формуле эллипса(x^2/a^2 +
    y^2/b^2 -1 = 0)
#параметры, которые будут подставляться в полученную формулу
B := ⟨0; 0⟩ :
    A := Matrix([[1 / a1^2, 0], [0, 1 / b1^2]]) :
    x := ⟨x1; y1⟩ :
    help1 := Matrix([[A, B], [Transpose(B), -1]]) :
    help2 := Matrix( [[diag(-1, -1), x], [Transpose(x), z - (Transpose(x).x)[1][1]]]) :
    Fi := Determinant(help1 + q * help2) :
    F_tr_dist := discrim(Fi, q) :
unassign('A','B','Fi','help1','help2','x') :


 #функция подсчета точного расстояния от точки до эллипса
 trueDistance := proc(x0, y0, a0, b0) :
      help := subs(x1 = x0, y1 = y0, a1 = a0, b1 = b0, F_tr_dist) :
      ans := help = 0 :
      t := fsolve(ans) :
      fu := diff(help, z) :
      for i from 1 to nops([t]) do
            if ((t[i] < root and t[i] > 0) or (i = 1) or (root < 0 and t[i] > 0 )) then
            root := t[i] :
            fi
      od:
      #проверка на кратность найденного корня
      if abs(subs(z = root, fu)) < 0.000000001 or root < 0 then
          root :='NaN':
      else
          root := evalf(sqrt(root)) :
      fi:
      RETURN(root);
 end:

# Дальше идут 3 функции подсчета приближенного расстояния.Аргумент x0 - вектор (x,y)
# координат точки.
# A - матрица квадратичной формы.


# расстояние по формуле dho(Harkery and O'Leary)
```

```
distanceHO := proc (x0, y0, a0, b0) :
    help := subs(x1 = x0, y1 = y0, a1 = a0, b1 = b0, F_tr_dist) :
    ans := sqrt( - coeff(help, z, 0) / coeff(help, z, 1)) :
    RETURN(Re(evalf(ans)))
end:

#расстояние по формуле ds(Sampson's distance)
distanceD1 :=proc(x0, A) :
    G := x → Transpose(x).A.x - 1 :
    d1 := abs(G(x0)) / (2 * sqrt((Transpose(A.x0).(A.x0)))) :
    RETURN(Re(evalf(d1))) :
end:
#experimental d4
distanceD4 :=proc(x0, A) :
    G := x → Transpose(x).A.x - 1 :
    d1 := abs(G(x0)) / (2 * sqrt((Transpose(A.x0).(A.x0)))) :
    k := (Transpose(A.x0).A.(A.x0)) :
    grad2 := (Transpose(A.x0).(A.x0)) :
    g := G(x0) :
    d2 := d1 * sqrt( 1 + (g * k) / ( 2 * ( (grad2)^2 )) + ((g^2 * k^2) / (2 * grad2^4))
    + ((5 * k^3 * g^3) / (8 * grad2^6)) ) :
    RETURN(Re(evalf(d2))) :

end:

findEllipsePoint := proc(x0, A) :
    temp := (Transpose(x0).A.x0 - 1) :
    myEquation := 0 = temp :
    answer := evalf(solve(myEquation)) :
    RETURN(answer) :
end:

getPositiveRoot := proc(roots) :
    for i from 1 to nops(roots) do
        if (roots[i] ≥ 0) then
            RETURN(roots[i]) :
        fi
    od:
end:

getNormalAtPoint := proc(x0, a, b) :
    answer := ⟨2 * x0[1] / a^2, 2 * x0[2] / b^2⟩ :
    answer := answer / norm(answer, 2) :
    RETURN(answer)
end:

getSkewPoint := proc(myPoint, myNormal, distribution) :
    answer := myPoint + myNormal * random[distribution](1) :
    RETURN(answer) :
end:
```

```
evaluateExperimentPointsForEllipse := proc(a, b, distribution, step) :
    A := ⟨1/a^2, 0; 0, 1/b^2⟩ :
    endpoint := convert(a/step + 1, rational) :
    answer := array(1..endpoint, 1..2, [ ]) :
    iterator := 1 :
    for i from 0 to a by step do
        ellipsePoint := ⟨ i, y⟩ :
        result_y := findEllipsePoint(ellipsePoint, A) :
        result_y := getPositiveRoot([result_y]) :
        ellipsePoint := ⟨ i, result_y ⟩ :
        myNormal := getNormalAtPoint(ellipsePoint, a, b) :
        tempPointHolder := getSkewPoint(ellipsePoint, myNormal, distribution) :
        answer[iterator, 1] := tempPointHolder[1] :
        answer[iterator, 2] := tempPointHolder[2] :
        iterator := iterator + 1 :
    od:
    RETURN(Matrix(answer)) :
end:

saveInExcel := proc(data, idName) :
    address := cat(idName, ".xls") :
    Export(Matrix(data), address) :
end:

doExperiment := proc(first, second) :
    WORKPATH := "C:\\Users\\Влад\\Documents\\Maple\\work_05032018\\" :
    for a from 1 to Dimension(first) do
        for b from 1 to Dimension(second) do
            data := evaluateExperimentPointsForEllipse(first[a], second[b], normald[0, 1],
    0.01) :
            answer := array(1..Dimension(data)[1], 1..6, [ ]) :
            for i from 1 to Dimension(data)[1] do
                answer[i, 1] := data[i, 1] :
                answer[i, 2] := data[i, 2] :
                answer[i, 3] := trueDistance(data[i, 1], data[i, 2], first[a], second[b]) :
                answer[i, 4] := distanceHO(data[i, 1], data[i, 2], first[a], second[b]) :
                A := ⟨1/first[a]^2, 0; 0, 1/second[b]^2⟩ :
                answer[i, 5] := distanceD1(⟨data[i, 1], data[i, 2]⟩, A) :
                answer[i, 6] := distanceD4(⟨data[i, 1], data[i, 2]⟩, A) :
            od:

              saveInExcel(answer, cat(WORKPATH, "data_", first[a], "_", second[b])) :
        od:
    od:
end:
```

```
function [ data ] = getData2DForStats( path )
mydata = xlsread(path);
sorted = [];
iter = 1;
for i=1:length(mydata)
    if (~isnan(mydata(i, 3))) && (mydata(i, 6) ~= 0)
        sorted = [sorted; zeros(1,3)];
        for j=3:5
            sorted(iter, j - 2) = abs(mydata(i, 3) - mydata↙
(i, j + 1))/mydata(i, 3);
        end
        iter = iter + 1;
    end
end
data = sorted;
end
```

statistics2D.m

```
all_a = [10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20];
all_b = [20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30,↙
31,32,33,34,35,36,37,38,39,40,41,42];
var_a = [10,11,12];
var_b = [28,29,30,31,32,33,34,35,36,37,38,39,40,41,42];
mmean = [];
mmin = [];
WORKPATH = 'C:↙
\\Users\\Влад\\Documents\\Maple\\work_05032018\\';
statData = [];
for i=1:length(var_a)
    for j=1:length(var_b)
        temp = getDataForStats([WORKPATH 'data_' num2str↙
(var_a(i)) '_', num2str(var_b(j))]);
        statData = cat(1,statData,temp);
    end
end
mmean = mean(statData)*100;
mmax = max(statData)*100;
```