Проект InMotion

Результат разработки алгоритма триангуляции TDoA на языке Python

Выполнил: Чашков М. С.

Оглавление

Общее описание алгоритма TDoA

Данный документ описываем используемое решение алгоритма Time Difference of Arrival TDoA. Работа выполнена с использованием [4], [5], [6], [7], [1], [2], [3].

TdoA один из наиболее популярных методов позиционирования. В отличие от метода Time of Arrival (ToA), этот метот не требует измерения времени отправки сообщения от «метки» до «анкера». Метод использует только время приема сигнала от метки каждым анкером и скорость света.

Зная время прием сигнала двумя разными точками, можно определить разницу расстояний от соответствующих анкеров до метки

$$\Delta d = c \cdot \Delta t \tag{1}$$

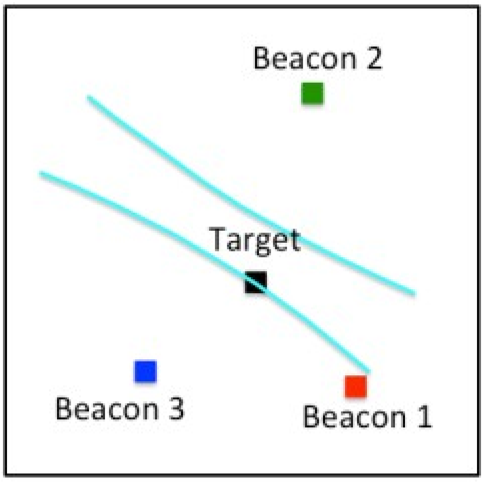
где c — скорость света

Δt – измеренная разность времени приема сигнала двумя анкерами

Гиперболой называется геометрическое место точек плоскости, модуль разности расстояний от каждой из которых до двух заданных точек F_1 и F_2 есть величина постоянная

В качестве двух заданных точек выступают анкеры.

Графический пример определения положения тела методом TDoA взят из [4] приведен на рисунке ниже



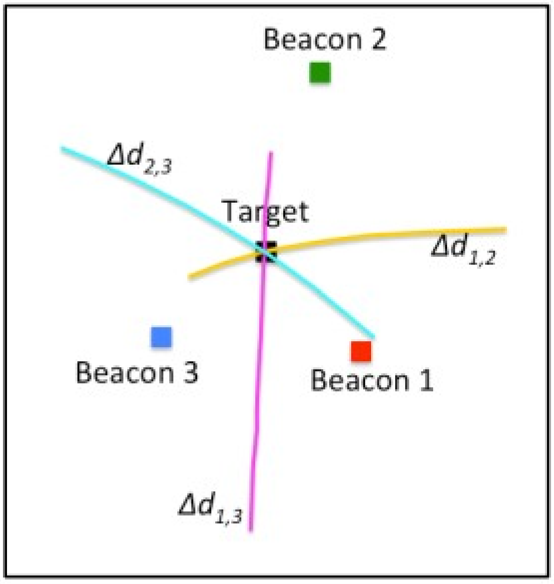


Рисунок 1 — Пример графического решения алгоритма TdoA

Для решения задачи необходимо составить алгоритм аналитического определения точки пересечения произвольного числа гипербол 3 и более анкеров.

Математическое описание алгоритма

Уравнения гипербол

Гипербола является, в общем случае кривой второго порядка. Общее уравнение которой может быть записано следующим образом:

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

С другой стороны гиперболу можно задать каноническим уравнением

$$\frac{x^2}{a^2}-\frac{y^2}{b^2}=1 \tag{3}$$

Для отдельной гиперболы введем в рассмотрение две системы:

1. Система координат поля (0XY)
2. система координат с началом координат в центре гиперболы (точка O) и осью OX сонаправленной с линией соединяющей фокусы (линия F_1F_2), как показано на рисунке 2

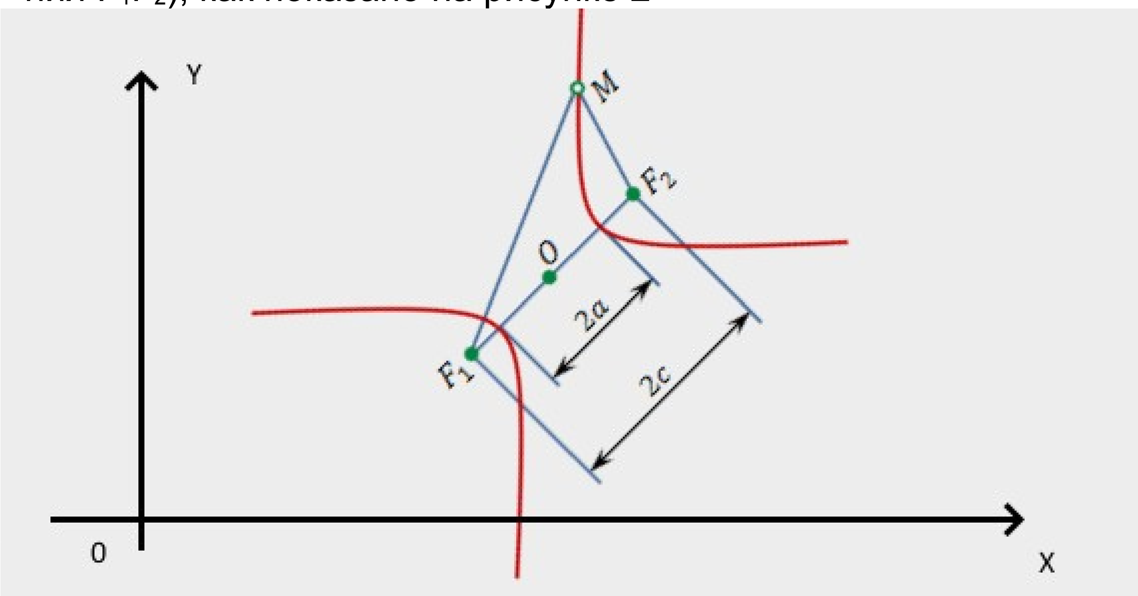


Рисунок 2 Гипербола в системе координат поля

Исходными данными для построения уравнения гиперболы являются:

1. измеренная разность расстояний от метки до каждого анкера Δd
2. координаты анкеров F_1 и F_2

В системе координат гипербол удобно использовать каноническое уравнение. При этом параметры легко определяются

$$
\begin{gathered}
|\Delta d| = 2a \\
c^2 = (F_{2x} - F_{1x})^2 + (F_{2y} - F_{1y})^2 \\
b^2 = c^2 - a^2
\end{gathered} \tag{4}
$$

Откуда легко вычисляются параметры общего уравнения кривой (2)

$$
\begin{gathered}
A_g = b^2 \\
B_g = 0 \\
C_g = -a^2 \\
D_g = 0 \\
E_g = 0 \\
F_g = -a^2 \cdot b^2
\end{gathered} \tag{5}
$$

Для того, чтобы записать уравнение гиперболы в системе координат поля необходимо выполнить две операции:

1. Поворот на угол Fi (угол от оси F_1F_2 к оси OX)
2. Параллельный перенос (смещение) на вектор [offset_x, offset_y]

как показано на рисунке 3

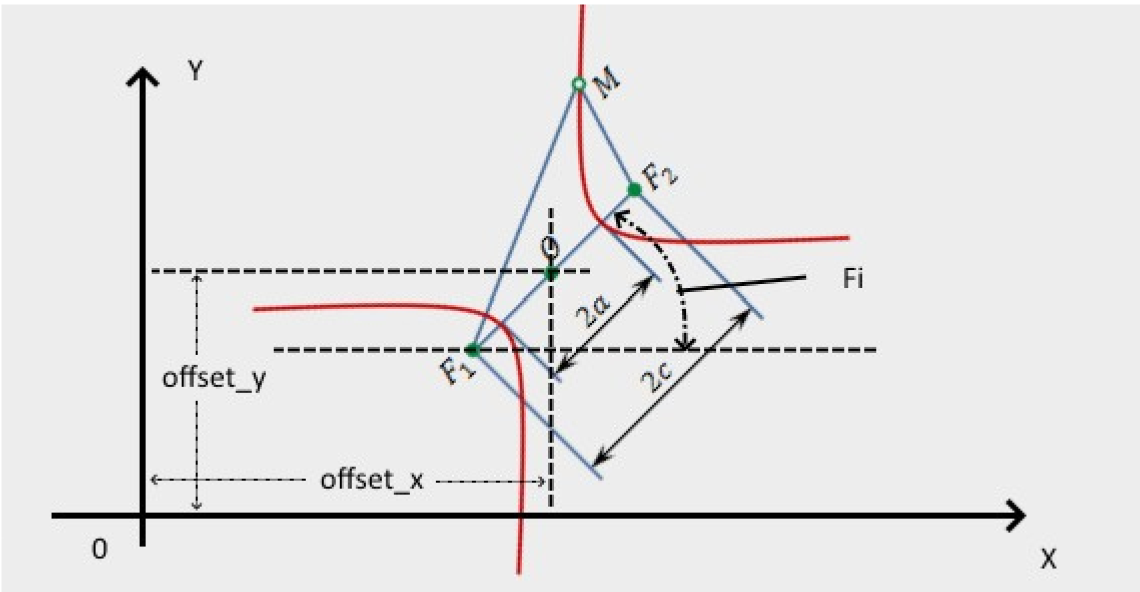


Рисунок 3 Параметры пересчета системы координат гиперболы

Для выполнения поворота используется матрица

$$
M(\Theta) = \begin{matrix} \cos(\Theta) & -\sin(\Theta) \\ \sin(\Theta) & \cos(\Theta) \end{matrix} \tag{6}
$$

В результате в системе координат поля параметры уравнения (2) можно определить следующим образом

$$\begin{gathered}A=b^2\cos^2(\Theta)-a^2\sin^2(\Theta)\\B=-2\cdot\sin(\Theta)\cos(\Theta)(a^2+b^2)\\C=b^2\sin^2(\Theta)-a^2\cos^2(\Theta)\\D=-2A\cdot offsetX-B\cdot offsetY\\E=-B\cdot offsetX-2C\cdot offsetY\\F=A\cdot offsetX^2+Bcdot\ offsetX\cdot offsetY+C\ offsetY^2-a^2\cdot b^2\end{gathered} \tag{7}$$

В общем случае на вход алгоритма подаются разницы измеренных разностей, в виде матрицы (в примере ниже для 4 анкеров)

$$\Delta=\begin{bmatrix}0 & \Delta_{1-2} & \Delta_{1-3} & \Delta_{1-4}\\ \Delta_{2-1} & 0 & \Delta_{2-3} & \Delta_{2-4}\\ \Delta_{3-1} & \Delta_{3-2} & 0 & \Delta_{3-4}\\ \Delta_{4-1} & \Delta_{4-2} & \Delta_{4-3} & 0\end{bmatrix} \tag{8}$$

Используя уравнения (7) составим матрицы для каждого из параметров

$$A=\begin{bmatrix}0 & A_{1-2} & A_{1-3} & A_{1-4}\\ A_{2-1} & 0 & A_{2-3} & A_{2-4}\\ A_{3-1} & A_{3-2} & 0 & A_{3-4}\\ A_{4-1} & A_{4-2} & A_{4-3} & 0\end{bmatrix} \tag{9}$$

$$B=\begin{bmatrix}0 & B_{1-2} & B_{1-3} & B_{1-4}\\ B_{2-1} & 0 & B_{2-3} & B_{2-4}\\ B_{3-1} & B_{3-2} & 0 & B_{3-4}\\ B_{4-1} & B_{4-2} & B_{4-3} & 0\end{bmatrix} \tag{10}$$

$$C=\begin{bmatrix}0 & C_{1-2} & C_{1-3} & C_{1-4}\\ C_{2-1} & 0 & C_{2-3} & C_{2-4}\\ C_{3-1} & C_{3-2} & 0 & C_{3-4}\\ C_{4-1} & C_{4-2} & C_{4-3} & 0\end{bmatrix} \tag{11}$$

$$D=\begin{bmatrix}0 & D_{1-2} & D_{1-3} & D_{1-4}\\ D_{2-1} & 0 & D_{2-3} & D_{2-4}\\ D_{3-1} & D_{3-2} & 0 & D_{3-4}\\ D_{4-1} & D_{4-2} & D_{4-3} & 0\end{bmatrix} \tag{12}$$

$$E=\begin{bmatrix} 0 & E_{1-2} & E_{1-3} & E_{1-4} \\ E_{2-1} & 0 & E_{2-3} & E_{2-4} \\ E_{3-1} & E_{3-2} & 0 & E_{3-4} \\ E_{4-1} & E_{4-2} & E_{4-3} & 0 \end{bmatrix} \qquad (13)$$

$$F=\begin{bmatrix} 0 & F_{1-2} & F_{1-3} & F_{1-4} \\ F_{2-1} & 0 & F_{2-3} & F_{2-4} \\ F_{3-1} & F_{3-2} & 0 & F_{3-4} \\ F_{4-1} & F_{4-2} & F_{4-3} & 0 \end{bmatrix} \qquad (14)$$

В общем случае для N анкеров размерность этих матриц будет NxN

Примечание 1: Размерность определяется не общим количеством анкеров, а количеством анкеров принявших сигнал от метки. Вполне реальная ситуация, при общем числе анкеров например 4 иметь размерность матриц 3x3.

Примечание 2: Минимальная размерность 3x3. Иначе невозможно определить координаты точки.

Проверка правильности расчетов проводилась для анкеров с координатами [0, 0] [0, 10] [15, 10]

Координаты метки [5, 3]

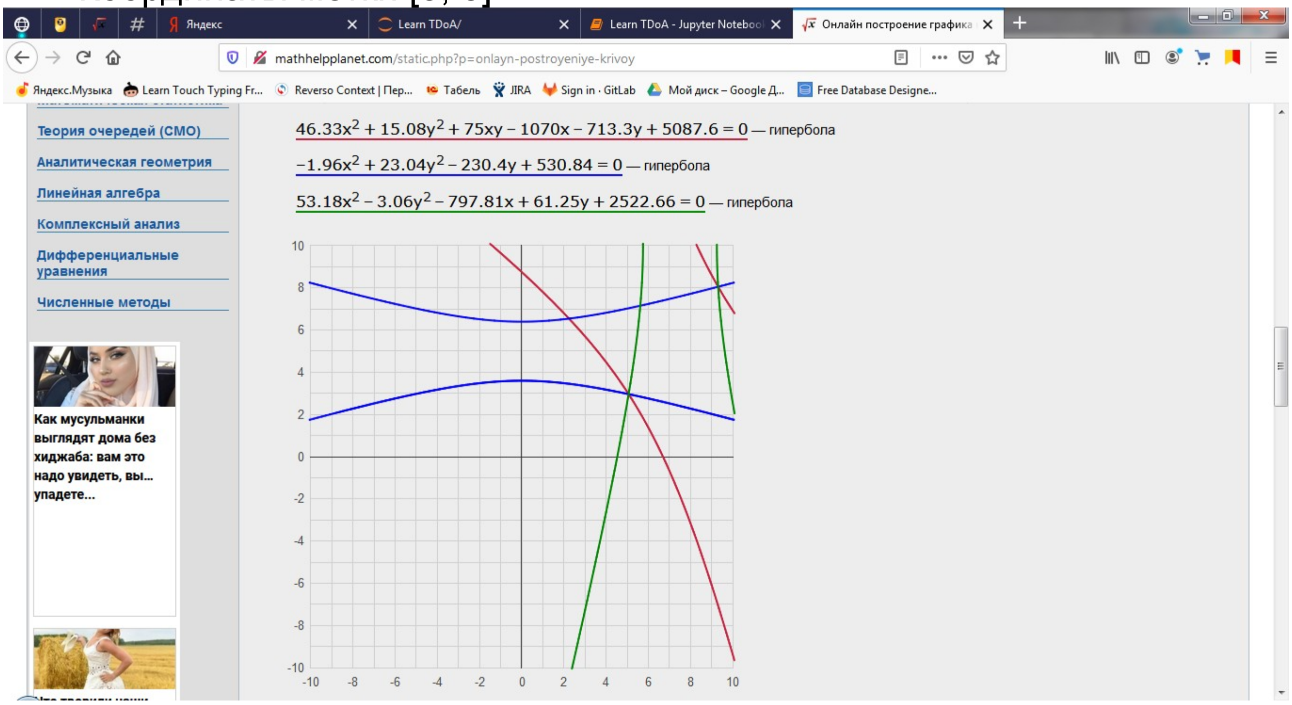


Рисунок 4 результат проверки

Алгоритм определения точки пересечения гипербол

Основная идея алгоритма заключается в том, что в точке пересечения разность уравненbй гипербол обращается в 0

Исходя из этого для трех анкеров имеем 3 гиперболы, для четырех анкеров уже 6 а в случае N анкеров число гипербол K определяется числом сочетаний из N по 2

$$K = C_N^2 \tag{15}$$

Дальше составляем систему уравнений для определения точки пересечения. Для трех анкеров и трех гипербол система будет выглядеть следующим образом

$$\begin{aligned} f_{1-2}(x,y) - f_{2-3}(x,y) = 0 = G_1(x,y) \\ f_{1-2}(x,y) - f_{1-3}(x,y) = 0 = G_2(x,y) \\ f_{2-3}(x,y) - f_{1-3}(x,y) = 0 = G_3(x,y) \end{aligned} \tag{16}$$

В общем случае L - число уравнений системы определяется числом сочетаний:

$$L = C_K^2 \tag{17}$$

Данная система уравнений нелинейная. Для ее решения применил алгоритм Ньютона

Алгоритм Ньютона

Система уравнений для гипер

Формула для нахождения решения определяется так

$$x^{(k+1)}=x^{(k)}-W^{-1}(x^{(k)})\cdot G(x^{(k)}) \qquad (18)$$

где

$$W(x)=\begin{bmatrix} \frac{\partial g_1(x)}{\partial x_1} & \cdots & \frac{\partial g_1(x)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_n(x)}{\partial x_n} & \cdots & \frac{\partial g_n(x)}{\partial x_n} \end{bmatrix} \qquad (19)$$

Введем обозначения

$$\Delta x^{(k)}=-W^{-1}(x^{(k)})\cdot G(x^{(k)}) \qquad (20)$$

Алгоритм Ньютона описывается так

1. Задать начальное приближение и малое число e – точность. Положить k=0
2. Решить систему линейных алгебраических уравнений относительно поправки $\Delta x^{(k)}$
3. Вычислить следующее приближение по формуле (18)
4. Если $\Delta^{(k+1)}=\max_i |x_i^{(k+1)}-x_i^{(k)}|\leqslant\epsilon$ процесс закончить иначе увеличить k и перейти к п. 2

Классический алгоритм следует доработать, а именно,

1. якобиан преобразуется к виду W(x,y)
2. Якобиан легко определить из системы уравнений (16) и уравнением гиперболы (2)

$$W(x,y)=\begin{bmatrix} \frac{\partial f_{12}(x,y)}{\partial x}-\frac{\partial f_{23}(x,y)}{\partial x} & \frac{\partial f_{12}(x,y)}{\partial y}-\frac{\partial f_{23}(x,y)}{\partial y} \\ \frac{\partial f_{12}(x,y)}{\partial x}-\frac{\partial f_{13}(x,y)}{\partial x} & \frac{\partial f_{12}(x,y)}{\partial y}-\frac{\partial f_{13}(x,y)}{\partial y} \\ \frac{\partial f_{23}(x,y)}{\partial x}-\frac{\partial f_{13}(x,y)}{\partial x} & \frac{\partial f_{23}(x,y)}{\partial y}-\frac{\partial f_{13}(x,y)}{\partial y} \end{bmatrix} \qquad (21)$$

3. Для удобства расчетов необходимо сформировать матрицы производных по каждой координате

$$dif_x(x,y)=\frac{\partial f_{1-2}}{\partial x}=2A\cdot x+B\cdot y+D= \tag{22}$$

$$=\begin{bmatrix} 0 & 2A_{1-2}x+B_{1-2}y+D_{1-2} & \cdots & 2A_{1-4}x+B_{1-4}y+D_{1-4} \\ 2A_{2-1}x+B_{2-1}y+D_{2-1} & 0 & \cdots & 2A_{2-4}x+B_{2-4}y+D_{2-4} \\ 2A_{3-1}x+B_{3-1}y+D_{3-1} & 2A_{3-2}x+B_{3-2}y+D_{3-2} & \cdots & 2A_{3-4}x+B_{3-4}y+D_{3-4} \\ 2A_{4-1}x+B_{4-1}y+D_{4-1} & 2A_{4-2}x+B_{4-2}y+D_{4-2} & \cdots & 0 \end{bmatrix} \tag{23}$$

$$dif_y(x,y)=\frac{\partial f_{1-2}}{\partial y}=2C\cdot y+B\cdot x+E= \tag{24}$$

$$=\begin{bmatrix} 0 & 2C_{1-2}y+B_{1-2}x+E_{1-2} & \cdots & 2C_{1-4}y+B_{1-4}x+E_{1-4} \\ 2C_{2-1}y+B_{2-1}X+E_{2-1} & 0 & \cdots & 2C_{2-4}y+B_{2-4}x+E_{2-4} \\ 2C_{3-1}y+B_{3-1}X+E_{3-1} & 2C_{3-2}y+B_{3-2}x+E_{3-2} & \cdots & 2C_{3-4}y+B_{3-4}x+E_{3-4} \\ 2C_{4-1}y+B_{4-1}x+E_{4-1} & 2C_{4-2}y+B_{4-2}x+E_{4-2} & \cdots & 0 \end{bmatrix} \tag{25}$$

4. Якобиан не является квадратным, а имеет размерность Lx2. Поэтому в формуле (20) необходимо использовать псевдоинверсную матрицу, которая легко определяется библиотекой python numpy.

Определение начальной точки алгоритма Ньютона

При работе алгоритма существенное влияние оказывает выбор начальной точки приближения методом Ньютона. Гиперболы имеют две ветки и парные ветки так же дают пересечение.

Идея алгоритма использовать вычисленную разность расстояний до анкеров не по модулю, а с учетом знака.

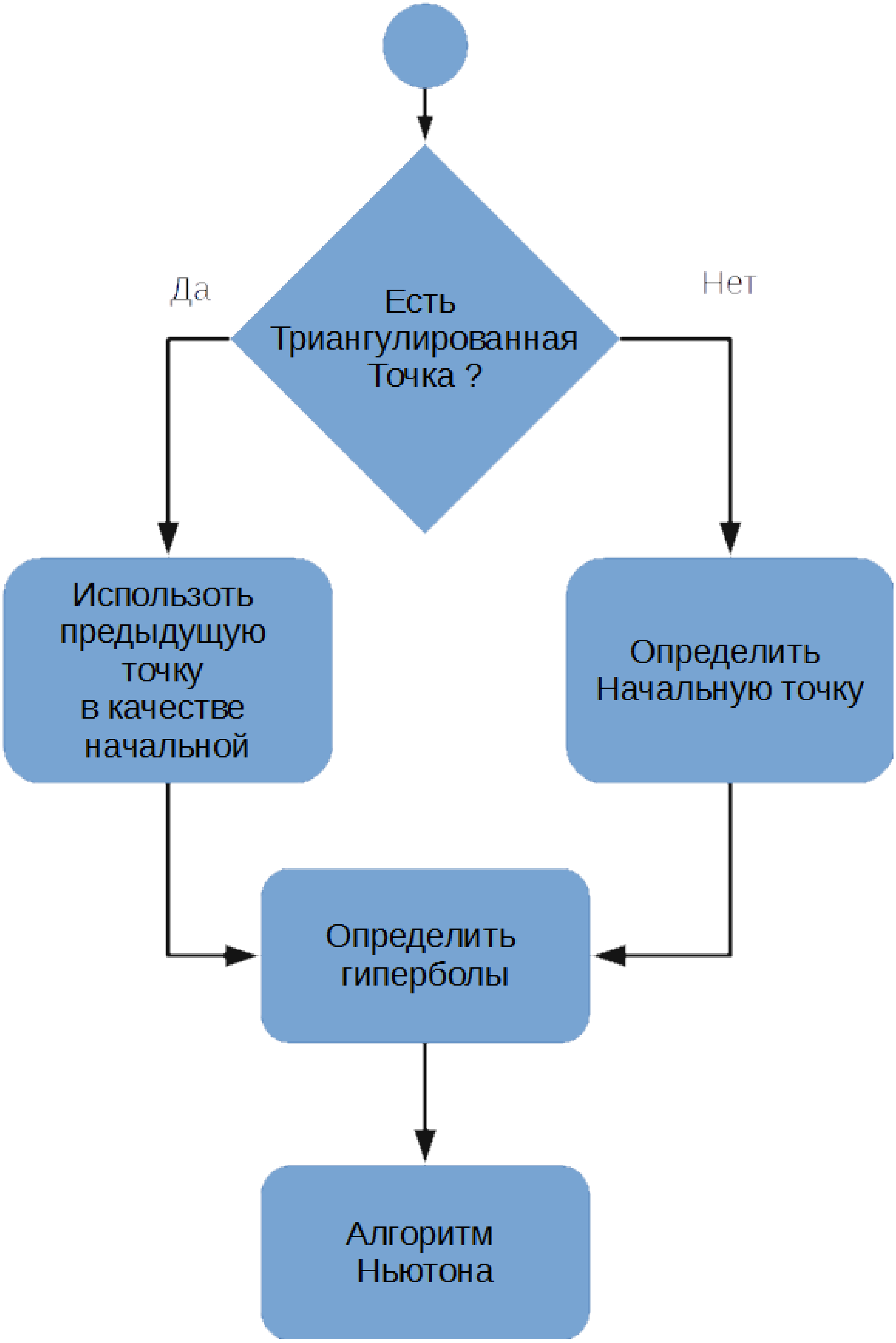
$$\Delta = \begin{bmatrix} 0 & \Delta_{1-2} & \Delta_{1-3} & \Delta_{1-4} \\ \Delta_{2-1} & 0 & \Delta_{2-3} & \Delta_{2-4} \\ \Delta_{3-1} & \Delta_{3-2} & 0 & \Delta_{3-4} \\ \Delta_{4-1} & \Delta_{4-2} & \Delta_{4-3} & 0 \end{bmatrix} \quad (26)$$

В формуле (26) , например, Δ_{1-2} может быть положительным, отрицательным или равным нулю. В первом случае в качестве начального приближения можно принять координаты анкера 2 во втором случае координаты анкера 1 а в третьем случае координаты середины отрезка соединяющего анкеры.

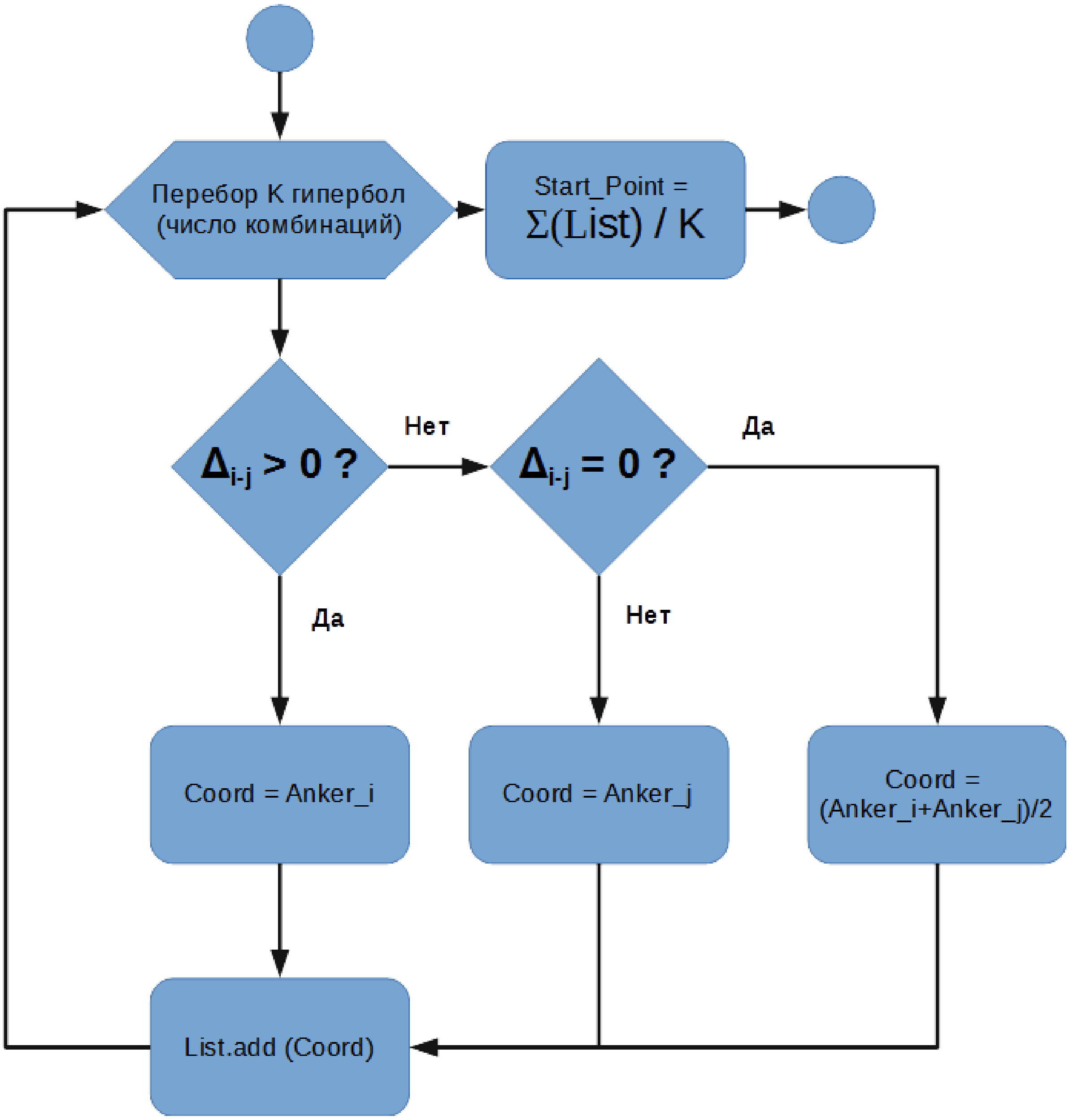
Зная общее число гипербол K из формулы (15) определяются K начальных приближений для каждой гиперболы, начальная точка определяется усреднением всех K приближений.

Структурные схемы алгоритма

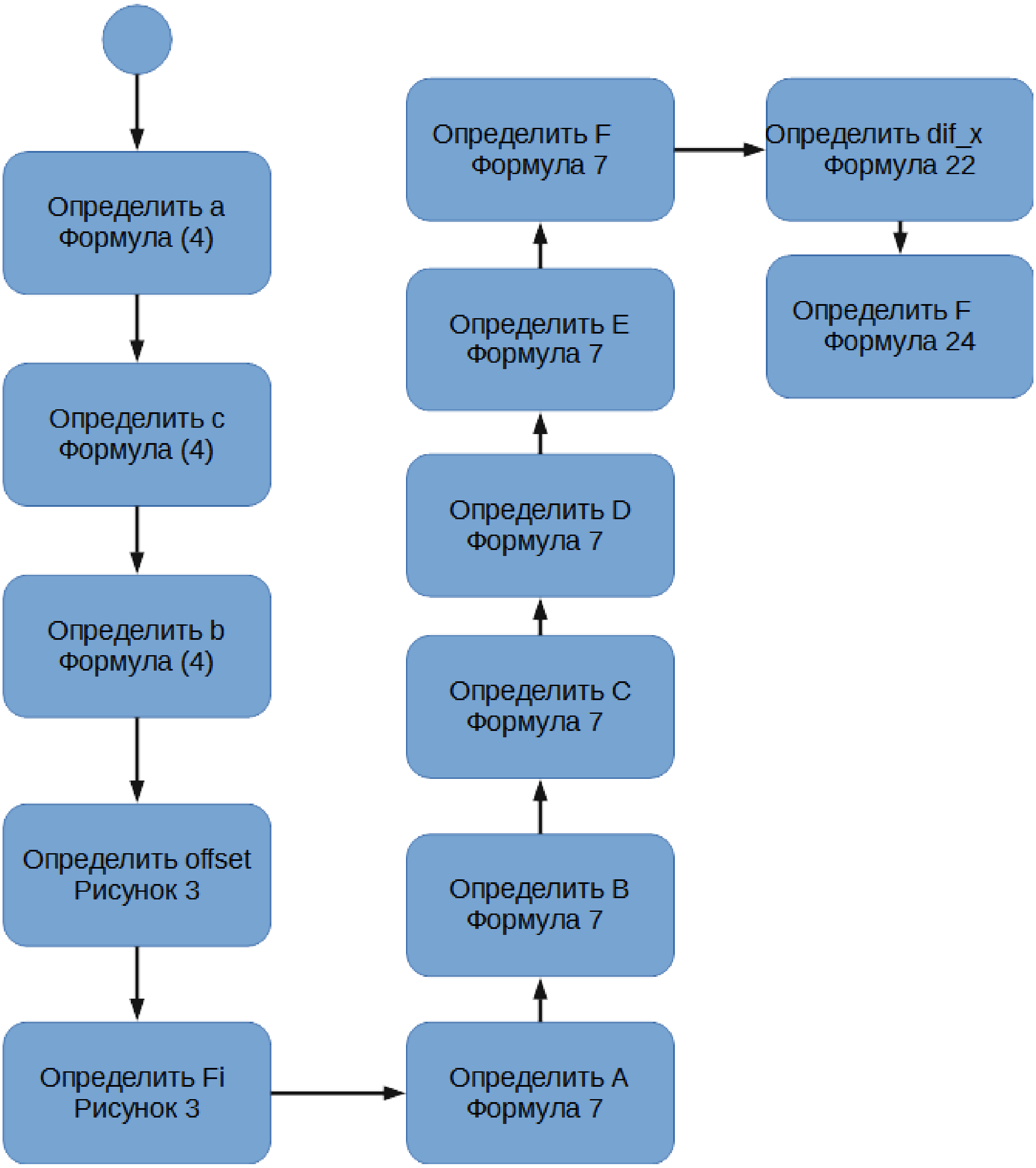
Общая схема алгоритма



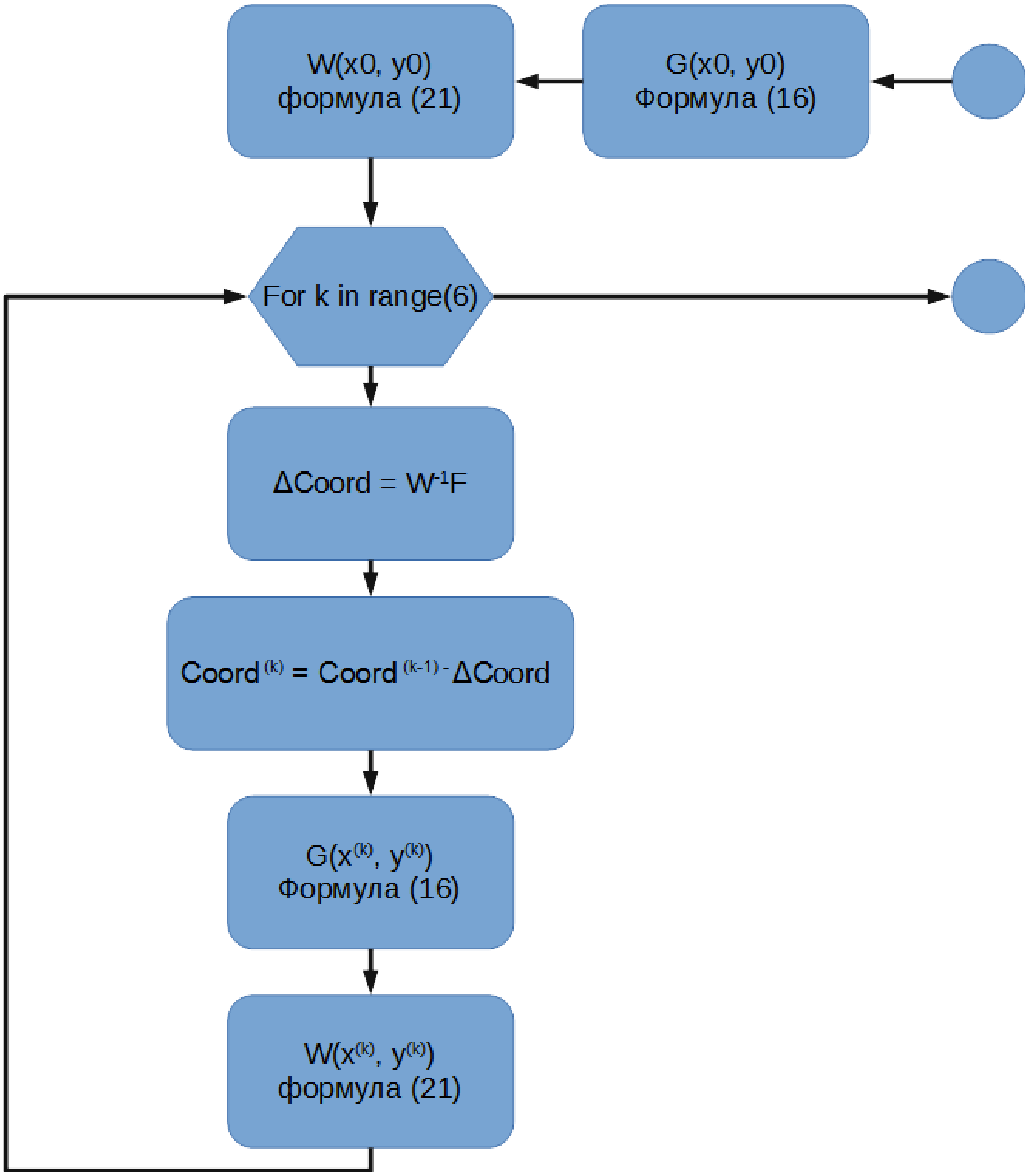
Определение начальной точки



Определение гипербол



Алгоритм Ньютона



Результаты работы алгоритма

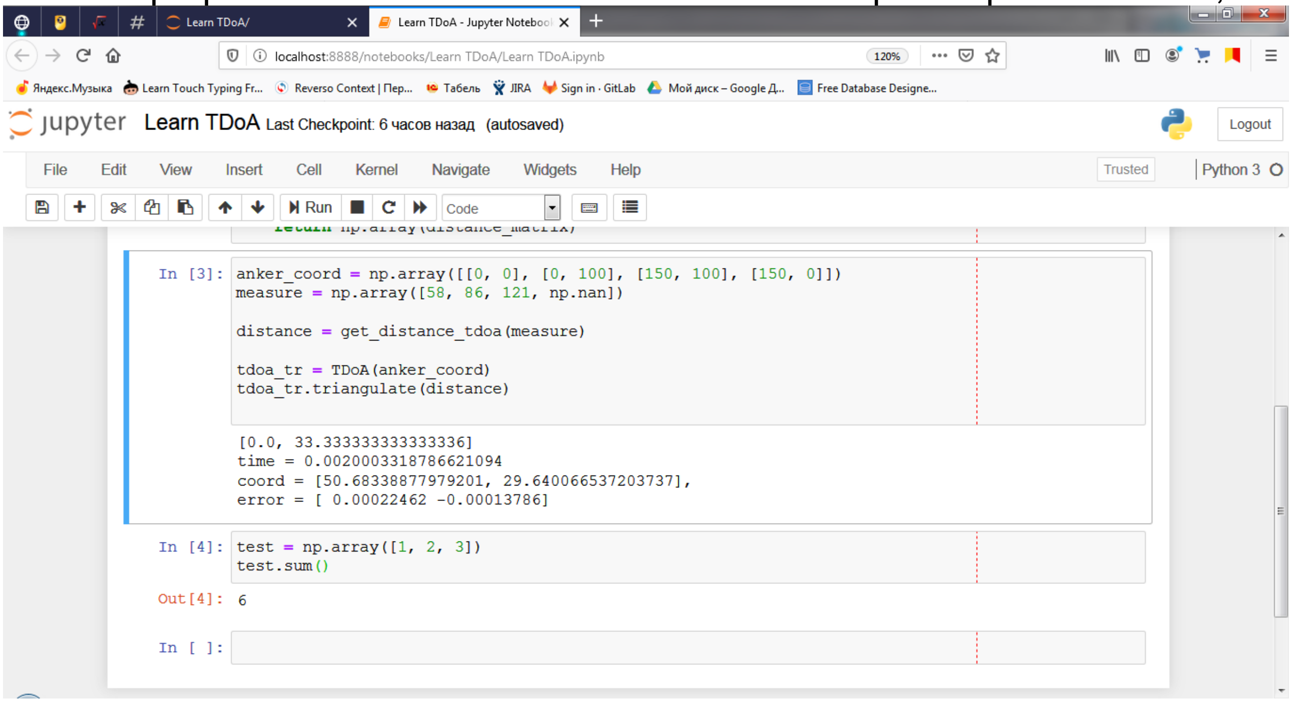
Общие результаты

Алгоритм сходится за 5-6 итераций.

Время определения одной точки составляет 3-10 мс

Точность определения точки при точном определении расстояний составляет меньше 10^{-4} м

На графиках ниже заложена неточность измерения расстояния 0,2 м



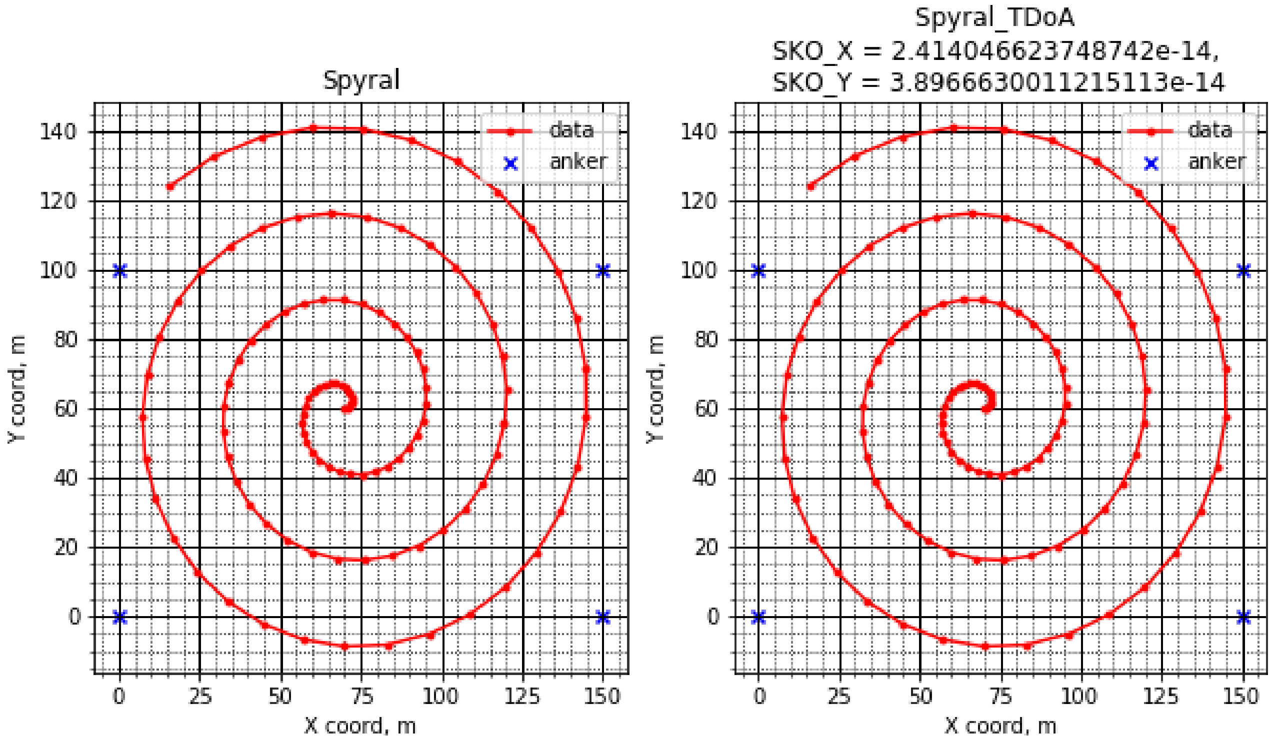
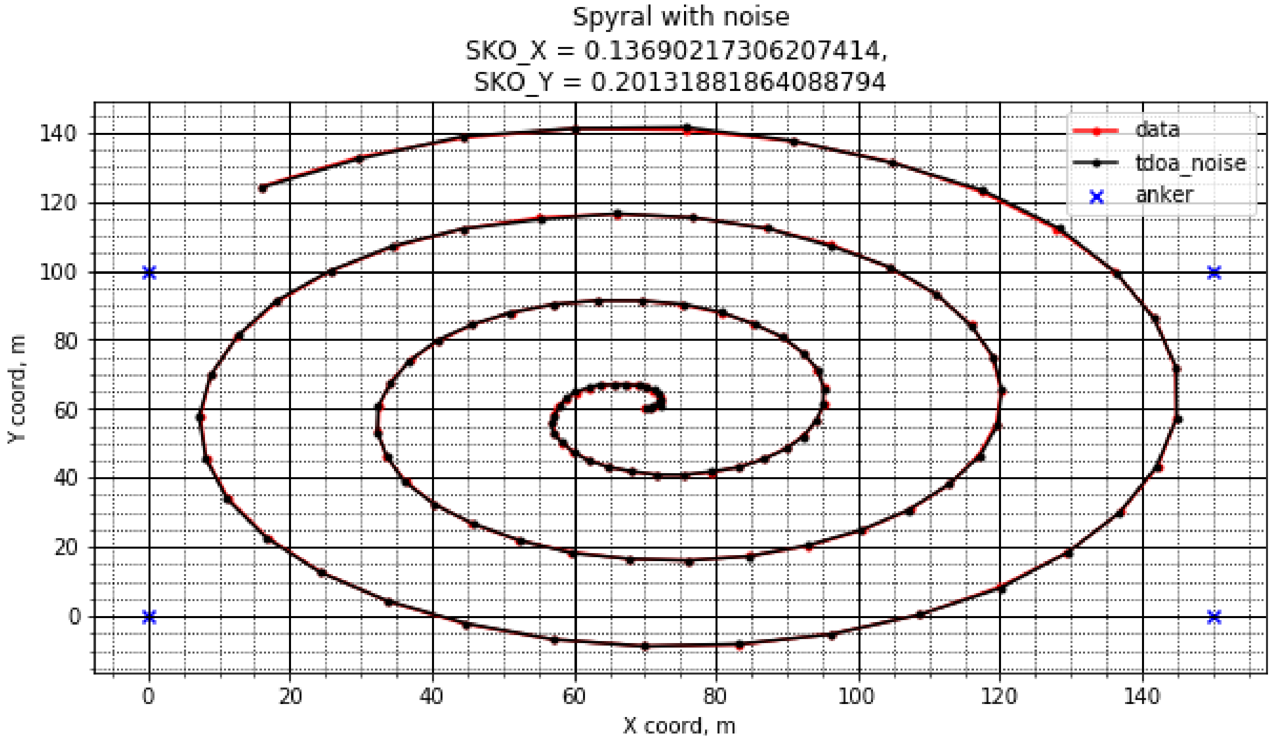
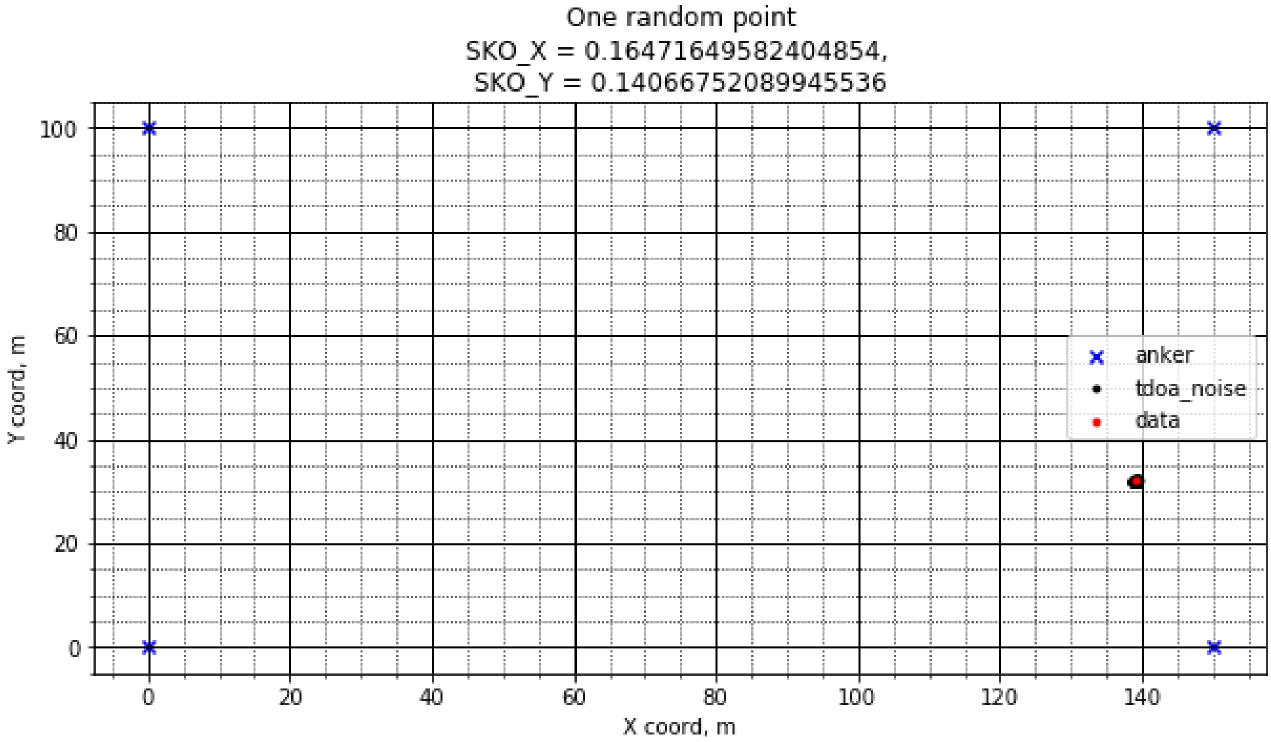
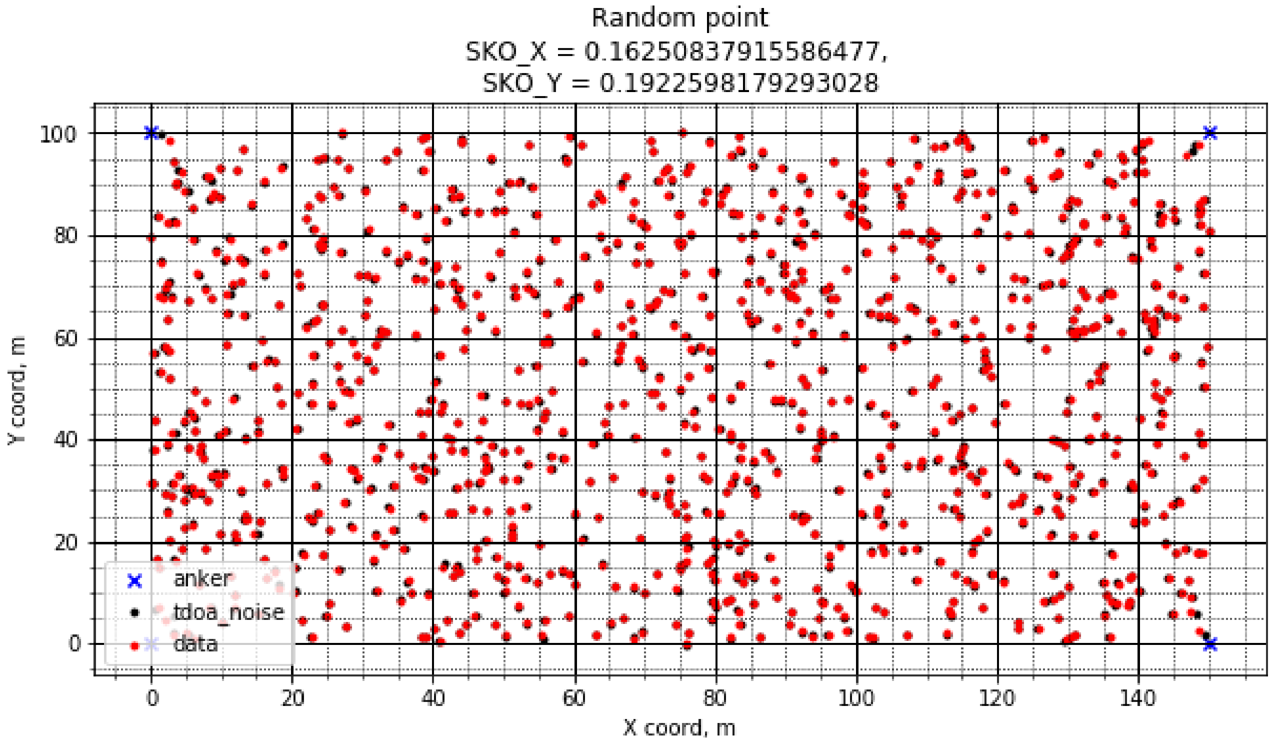


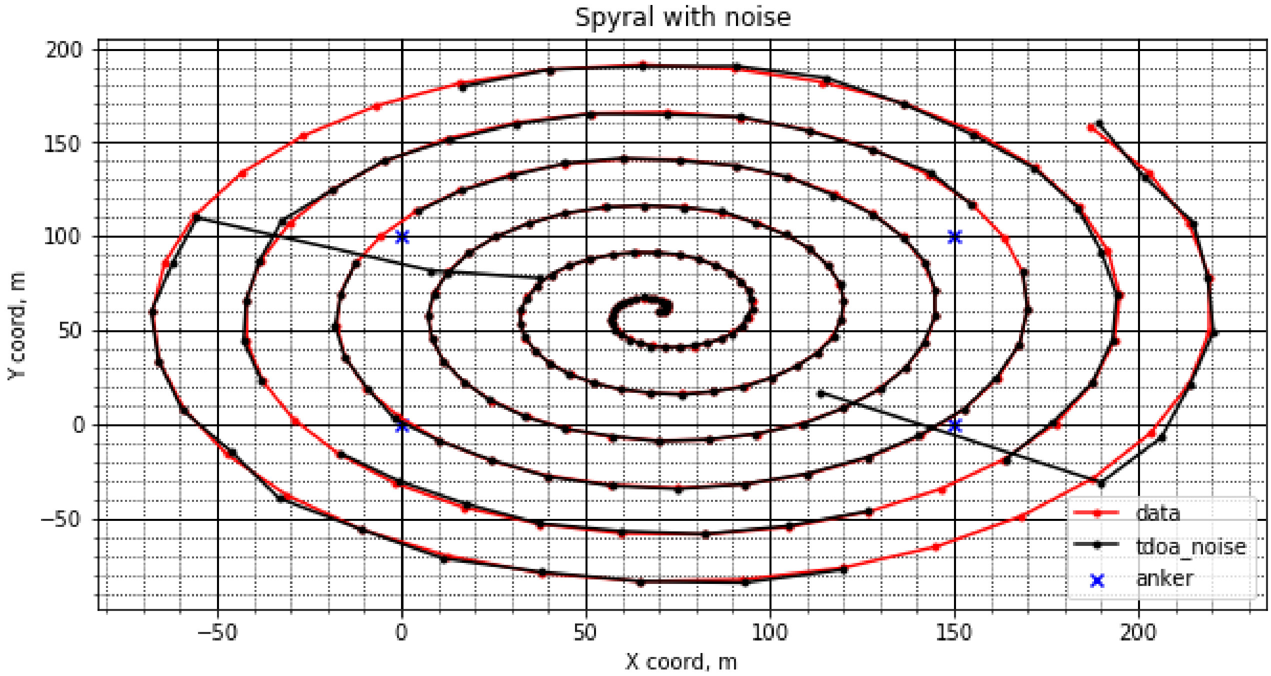
рисунок Триангуляция при точно измеренных расстояниях







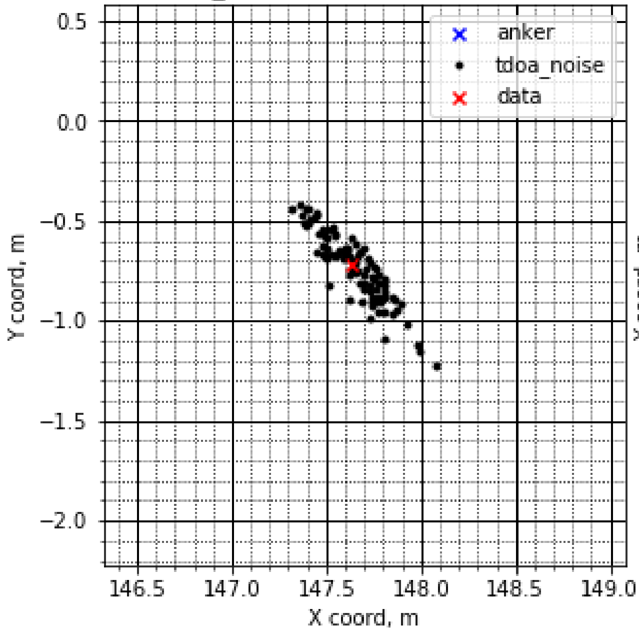
Есть особенность триангуляции за пределами поля, связанная с выбором начальной точки. При зашумленных данных возможно потеря триангуляции, в связи с невозможностью построения гиперболы через зашумленные данные. Результат может быть например таким


Spyral with noise
Y coord, m
X coord, m
data
tdoa_noise
anker
200
150
100
50
0
−50
−50
0
50
100
150
200

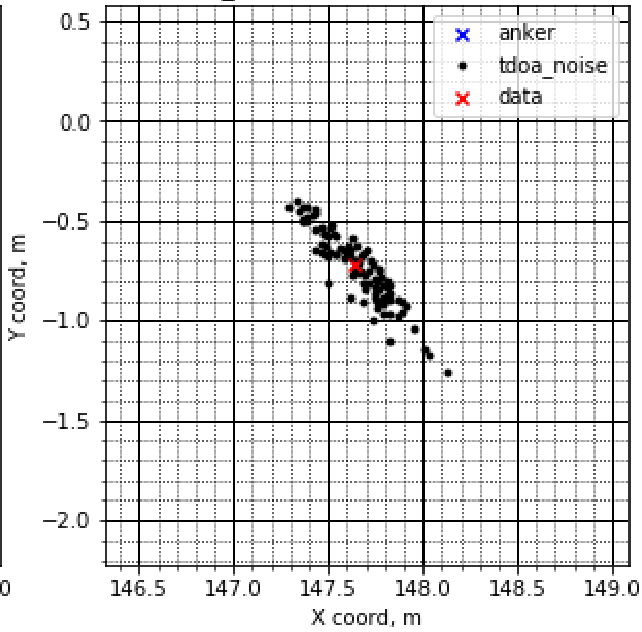
Тестирование работы алгоритма при неточных измерениях

Результаты тестирования на графиках ниже

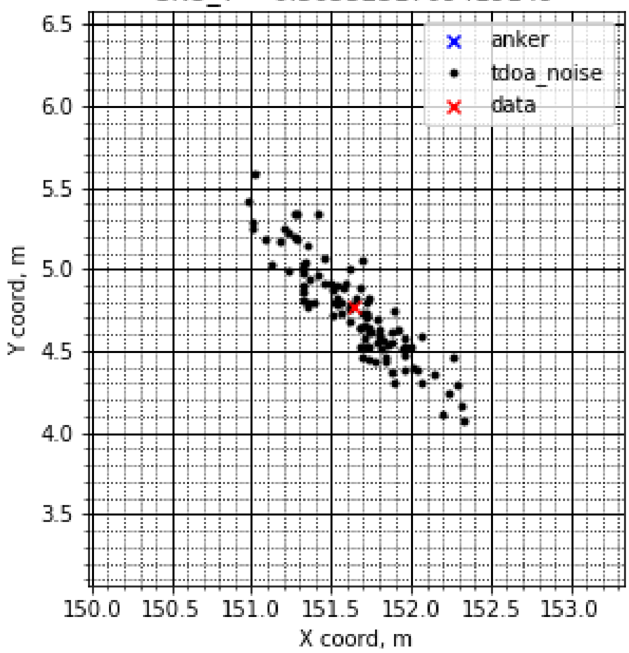
3 Anker use
Noise standart deviation = 0.05m
SKO_X = 0.15678033479336317,
SKO_Y = 0.17012660035350452



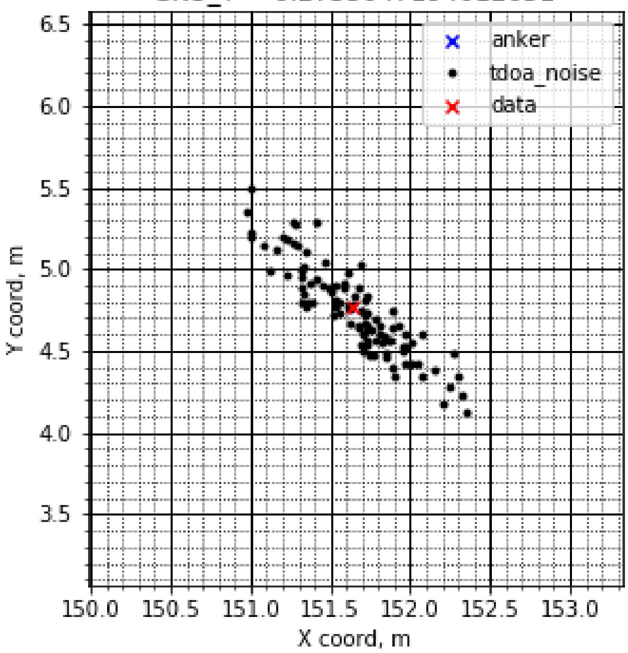
4 Anker use
Noise standart deviation = 0.05m
SKO_X = 0.17131086430903048,
SKO_Y = 0.1781747642662884

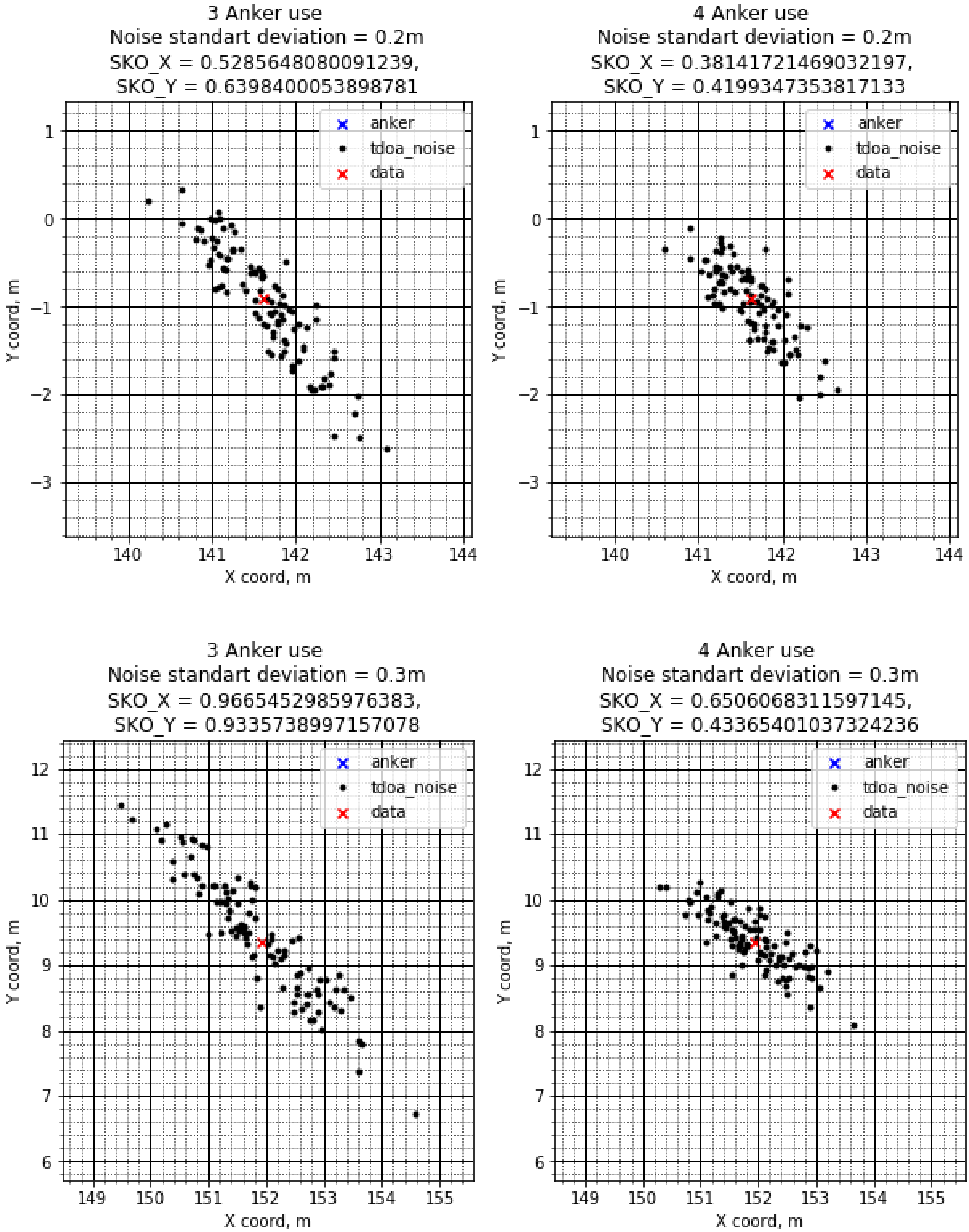


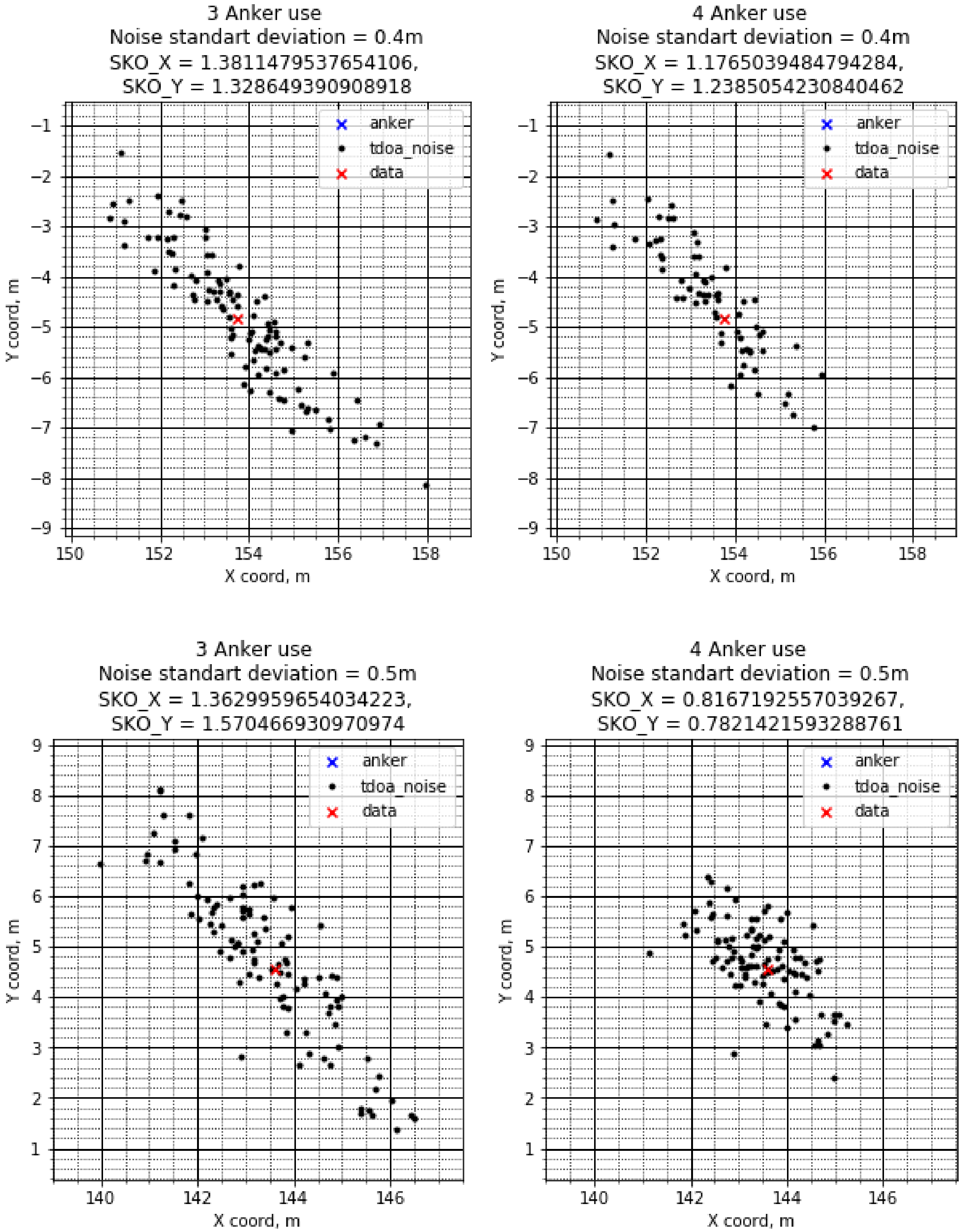
3 Anker use
Noise standart deviation = 0.1m
SKO_X = 0.30932536670239974,
SKO_Y = 0.3053231709419149

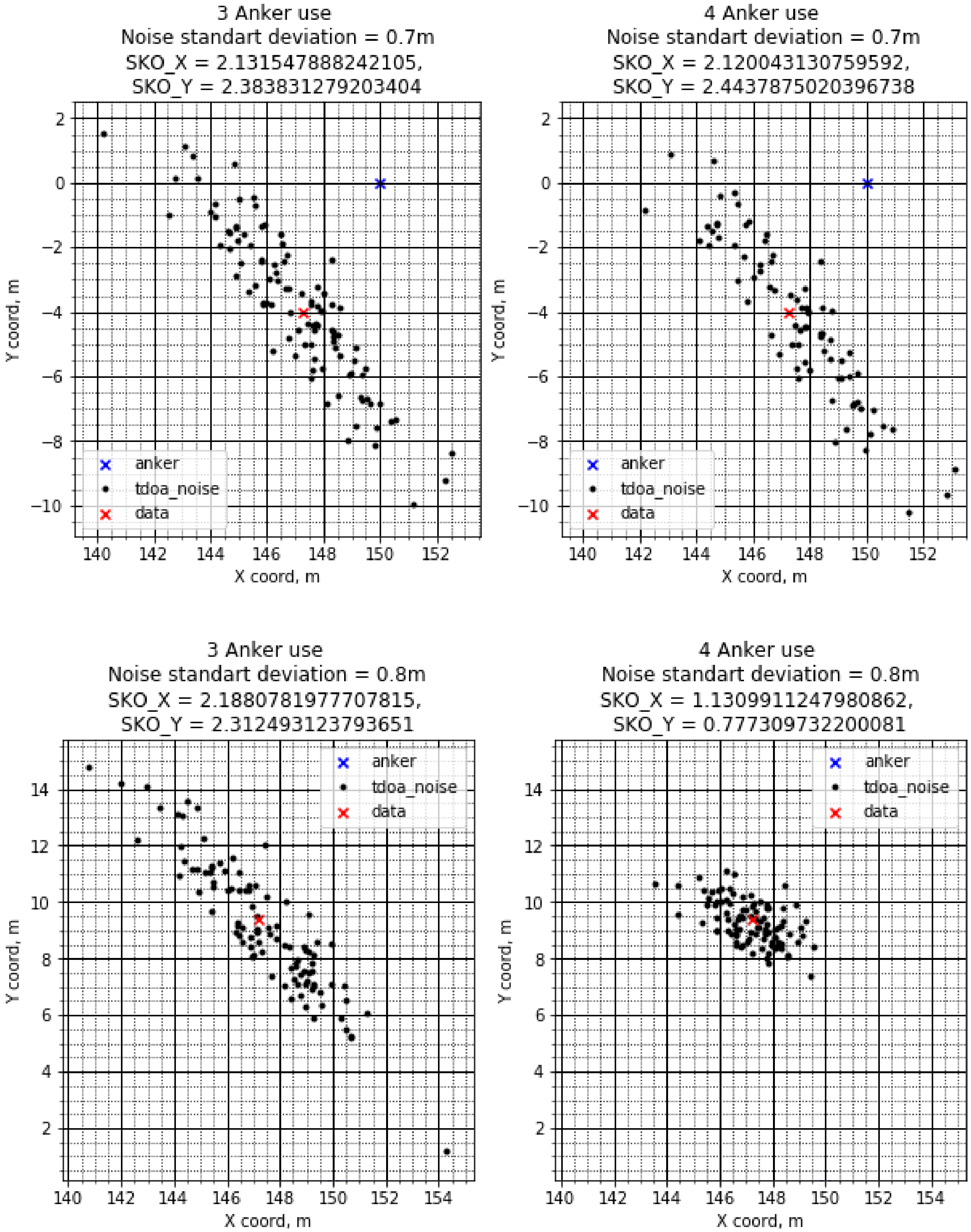


4 Anker use
Noise standart deviation = 0.1m
SKO_X = 0.31389494612505553,
SKO_Y = 0.2755047194012051

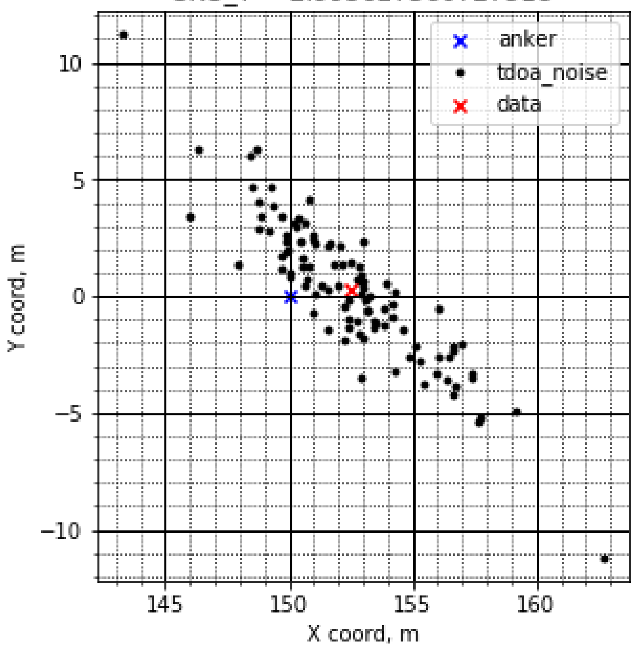



3 Anker use
Noise standart deviation = 0.2m
SKO_X = 0.5285648080091239,
SKO_Y = 0.6398400053898781
anker
tdoa_noise
data
Y coord, m
X coord, m
4 Anker use
Noise standart deviation = 0.2m
SKO_X = 0.38141721469032197,
SKO_Y = 0.4199347353817133
anker
tdoa_noise
data
Y coord, m
X coord, m
3 Anker use
Noise standart deviation = 0.3m
SKO_X = 0.9665452985976383,
SKO_Y = 0.9335738997157078
anker
tdoa_noise
data
Y coord, m
X coord, m
4 Anker use
Noise standart deviation = 0.3m
SKO_X = 0.6506068311597145,
SKO_Y = 0.43365401037324236
anker
tdoa_noise
data
Y coord, m
X coord, m

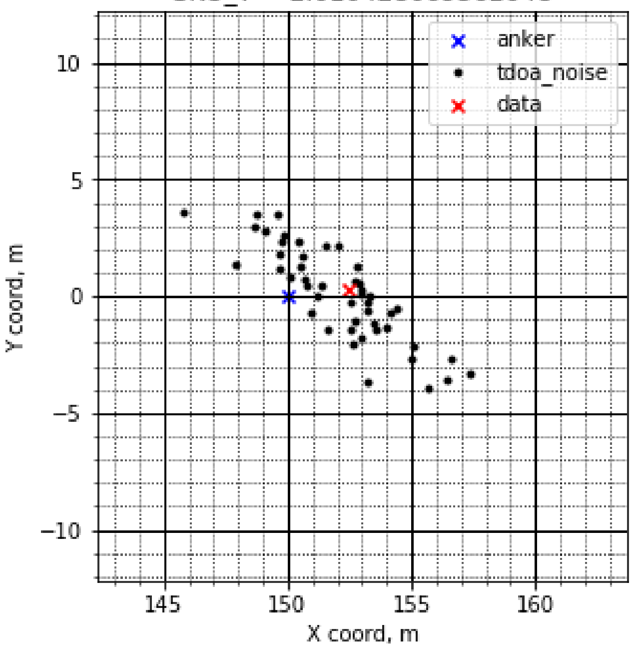

3 Anker use
Noise standart deviation = 0.4m
SKO_X = 1.3811479537654106,
SKO_Y = 1.328649390908918
anker
tdoa_noise
data
Y coord, m
X coord, m
4 Anker use
Noise standart deviation = 0.4m
SKO_X = 1.1765039484794284,
SKO_Y = 1.2385054230840462
anker
tdoa_noise
data
Y coord, m
X coord, m
3 Anker use
Noise standart deviation = 0.5m
SKO_X = 1.3629959654034223,
SKO_Y = 1.570466930970974
anker
tdoa_noise
data
Y coord, m
X coord, m
4 Anker use
Noise standart deviation = 0.5m
SKO_X = 0.8167192557039267,
SKO_Y = 0.7821421593288761
anker
tdoa_noise
data
Y coord, m
X coord, m


3 Anker use
Noise standart deviation = 0.7m
SKO_X = 2.131547888242105,
SKO_Y = 2.383831279203404
anker
tdoa_noise
data
Y coord, m
X coord, m
4 Anker use
Noise standart deviation = 0.7m
SKO_X = 2.120043130759592,
SKO_Y = 2.4437875020396738
anker
tdoa_noise
data
Y coord, m
X coord, m
3 Anker use
Noise standart deviation = 0.8m
SKO_X = 2.1880781977707815,
SKO_Y = 2.312493123793651
anker
tdoa_noise
data
Y coord, m
X coord, m
4 Anker use
Noise standart deviation = 0.8m
SKO_X = 1.1309911247980862,
SKO_Y = 0.777309732200081
anker
tdoa_noise
data
Y coord, m
X coord, m

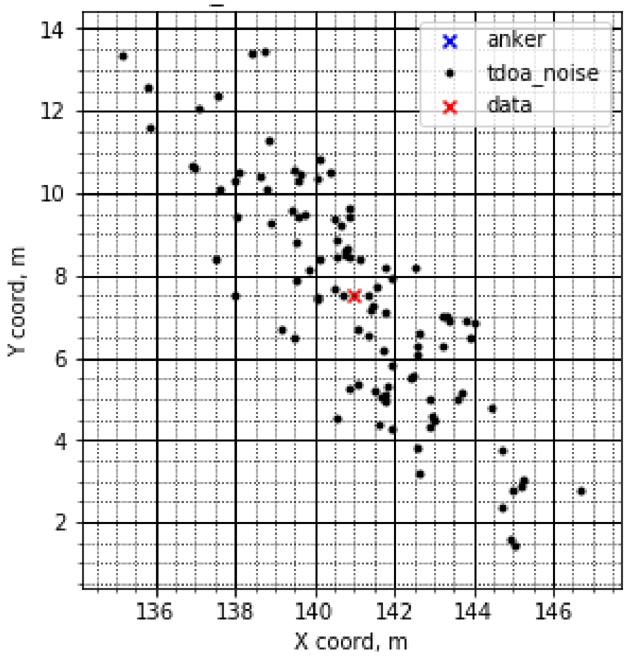
3 Anker use
Noise standart deviation = 0.9m
SKO_X = 2.986265281176649,
SKO_Y = 2.995627560727518



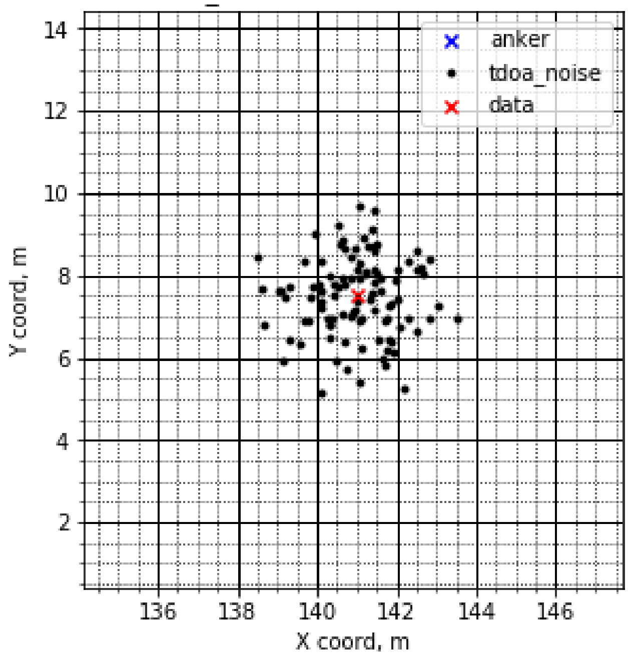
4 Anker use
Noise standart deviation = 0.9m
SKO_X = 2.3920129429610926,
SKO_Y = 2.010428669362048

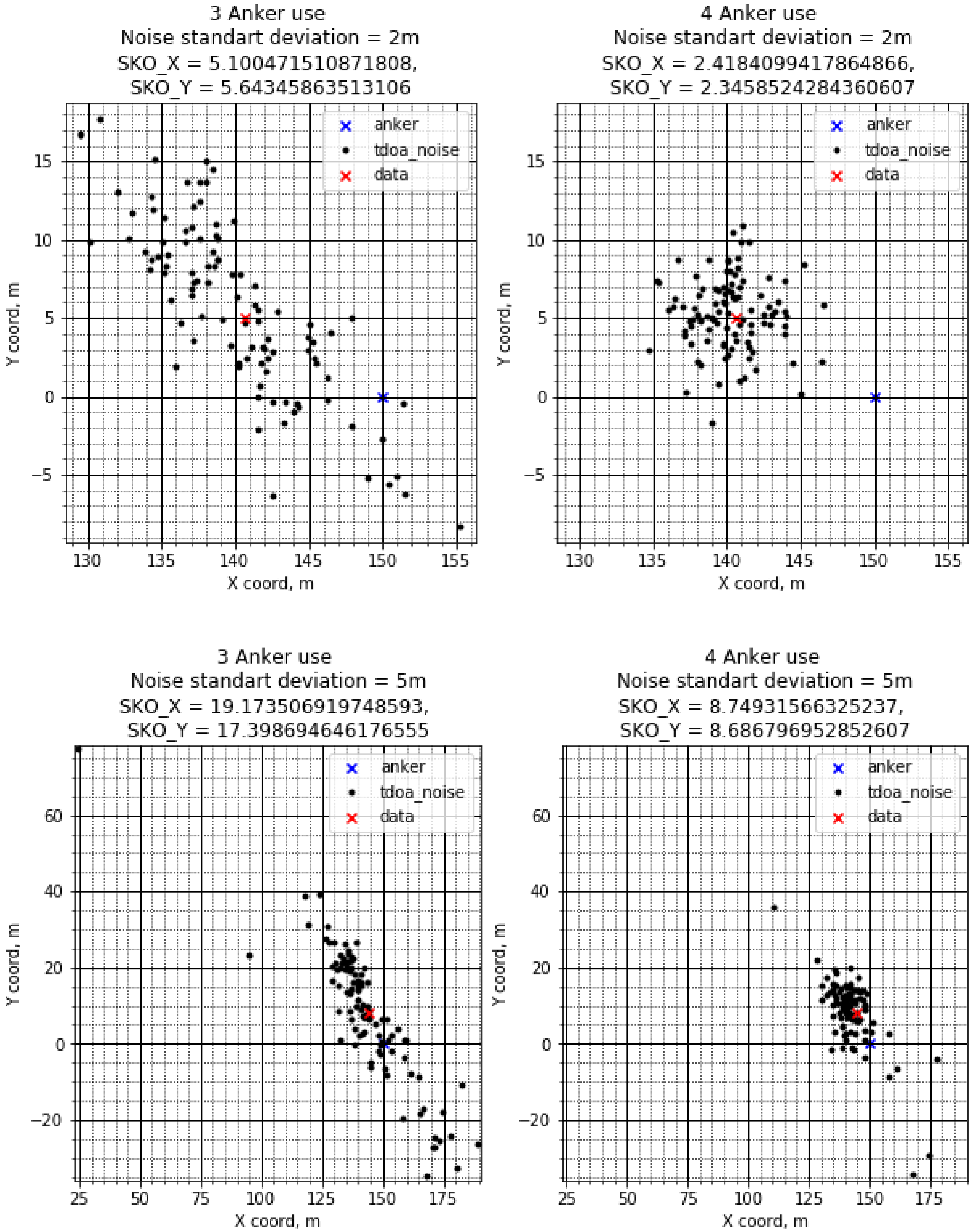


3 Anker use
Noise standart deviation = 1m
SKO_X = 2.29285370776433,
SKO_Y = 2.722528818886656

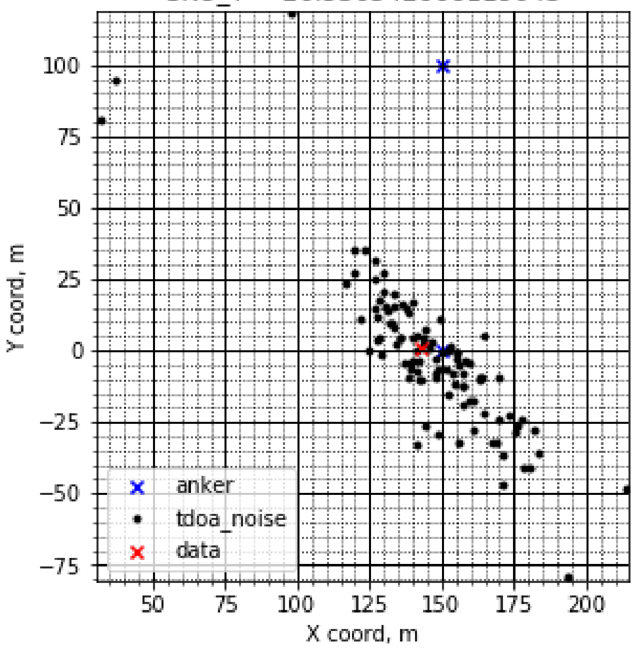


4 Anker use
Noise standart deviation = 1m
SKO_X = 1.0289062249266123,
SKO_Y = 0.9580825419740944

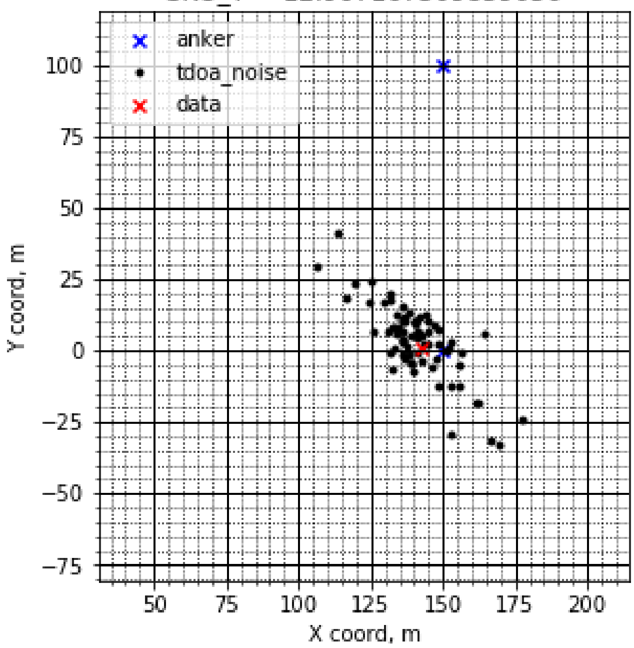



3 Anker use
Noise standart deviation = 2m
SKO_X = 5.100471510871808,
SKO_Y = 5.64345863513106
anker
tdoa_noise
data
Y coord, m
X coord, m
4 Anker use
Noise standart deviation = 2m
SKO_X = 2.4184099417864866,
SKO_Y = 2.3458524284360607
anker
tdoa_noise
data
Y coord, m
X coord, m
3 Anker use
Noise standart deviation = 5m
SKO_X = 19.173506919748593,
SKO_Y = 17.398694646176555
anker
tdoa_noise
data
Y coord, m
X coord, m
4 Anker use
Noise standart deviation = 5m
SKO_X = 8.74931566325237,
SKO_Y = 8.686796952852607
anker
tdoa_noise
data
Y coord, m
X coord, m

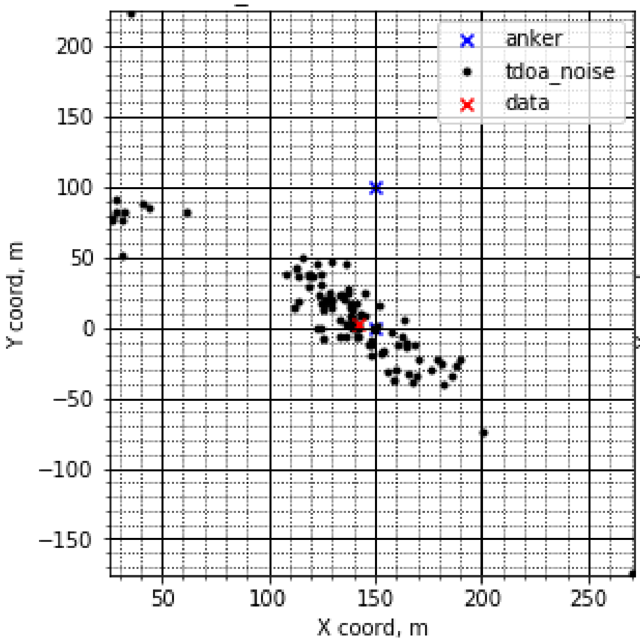
3 Anker use
Noise standart deviation = 7m
SKO_X = 24.511515933616113,
SKO_Y = 26.536542066119043



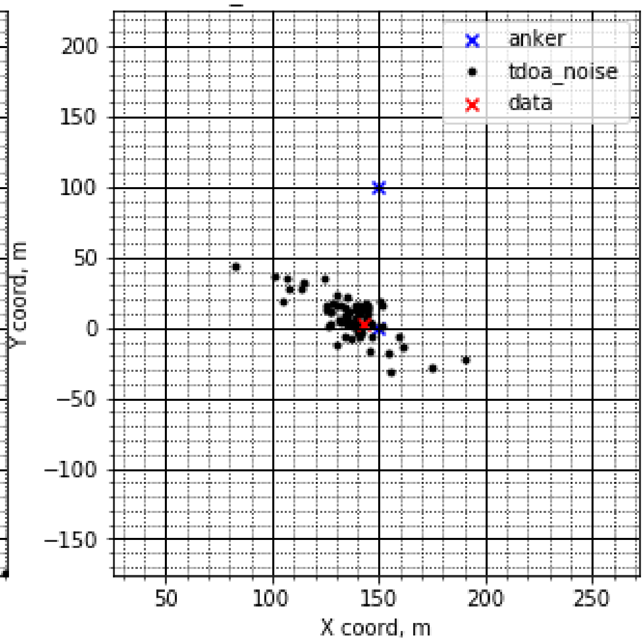
4 Anker use
Noise standart deviation = 7m
SKO_X = 12.255144514826567,
SKO_Y = 12.907167509859056



3 Anker use
Noise standart deviation = 10m
SKO_X = 42.24987491877377,
SKO_Y = 43.81390375763038



4 Anker use
Noise standart deviation = 10m
SKO_X = 16.449119764367413,
SKO_Y = 15.386401841572999



Выводы

1. СКО погрешности триангулирования составляет примерно 3σ от погрешности измерения расстояний
2. Использования 4 анкеров для триангуляции в общем случае дает более точное позиционирование
3. При СКО шумов измерений равном 4 м и более возникают достаточно значительные отрывы триангулированных точек.

Библиография

1: Методы решения систем нелинейных уравнений, , https://docs.google.com/presentation/d/12vkhVUSlikrB5D0oPjlqu06cfvEQEHANRZzHQ79ir6w/edit#slide=id.g858496cd6e_0_0

2: Массивы NumPy , , https://pyprog.pro/introduction.html

3: Гипербола: определение, свойства, построение, , http://mathhelpplanet.com/static.php?p=giperbola

4: Brian O'Keefe, Finding Location with Time of Arrival and Time Difference of Arrival Techniques, 2017

5: Fatima S. Al Harbi, Hermann J. Helgert,, An Improved Chan-Ho Location Algorithm for TDOA Subscriber Position Estimation, 2010

6: Jianghuai Pan, A New Robust Multi-station TDOA Localization Algorithm, 2017

7: Wei WANG, Junjie HUANG, Shaobin CAI , Junjie YANG, Design and Implementation of Synchronization-freeTDOA Localization System Based on UWB, 2018

Приложение А Код алгоритма реализованный на языке Python

Код алгоритма доступен по ссылке: https://gitlab.fablite.tech/klimenko.as/inmotion_pyproj/-/commit/6b0f94ec22c327a9134f45fc3a437a2b3d1e6265