Математические структуры и моделирование 2014. №2(30). С. 62–76

УДК 004.942:004.732:621.396.969

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ ОЦЕНИВАНИЯ КООРДИНАТ ТОЧКИ БЕСПРОВОДНОГО ДОСТУПА ПО ИЗМЕРЕНИЯМ МОЩНОСТИ ПРИНИМАЕМЫХ СИГНАЛОВ

Д.Н. Лавров[1]
к.т.н., зав. каф. компьютерных технологий и сетей, e-mail: lavrov@omsu.ru
О.А. Вишнякова[3]
инженер-программист, e-mail: olga@infotekorg.ru
Е.И. Дудяк[2]
инженер-программист, e-mail: gggorec@gmail.com
С.Ю. Лаврова[1]
магистр информатики и вычислительной техники, e-mail: Sveta.Lavrova@gmail.com

[1]Факультет компьютерных наук, Омский государственный университет
[2]Омский научно-исследовательский институт приборостроения
[3]ООО «Аспект»

Аннотация. В работе представлен программный код и результаты компьютерного моделирования определения координат беспроводной точки доступа. В предыдущей работе была построена математическая модель распределённого измерителя. Данная работа посвящена проверке работоспособности и качества разработанных алгоритмов. Сравнительному анализу были подвержены две целевые функции. По результатам компьютерного моделирования сделан вывод о том, что метод, основанный на целевой функции, минимизирующей невязку системы уравнений, предпочтительнее.

Ключевые слова: точки доступа, беспроводные технологии, радиолокация, метод Ньютона-Рафсона, компьютерное моделирование.

1. Постановка задачи

В работе [1] была предложена вычислительная схема на основе метода Ньютона-Рафсона для определения координат точки беспроводного доступа на основе измерения мощности сигнала точки доступа с разнесённых в пространстве точек наблюдения. В работе были предложены две целевые функции:

$$||\eta||^2 = F(x,y,n) = \sum_{i=1}^{n} \left((x-x_i)^2 + (y-y_i)^2 - R_i^2\right)^2 \to \min \tag{1}$$

и

$$||\varepsilon||^2 = G(x,y,n) = \sum_{i=1}^{n} \left(\frac{(x-x_i)^2 + (y-y_i)^2 - R_i^2}{2R_i}\right)^2 \to \min, \tag{2}$$

минимизация которых по аргументам x и y даёт нам оценку координат. Здесь η — невязка системы уравнений; ε — ошибка измерения расстояний до точки доступа (радиусов); n — число измерений; (x_i, y_i) — координаты i-й точки наблюдения; R_i — оценка по мощности излучения расстояния от точки наблюдения до точки доступа; (x, y) — искомые координаты точки беспроводного доступа.

Далее в работе [1] приводится алгоритм, реализующий определение оценок координат:

Шаг 0. Для первых трёх поступивших измерений вычисляем начальное приближение по алгоритму [1, стр. 52-54]. Положим $n = 3$.

Шаг 1. Уточняем решение методом Ньютона-Рафасона по формулам

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \end{pmatrix} - \left(\begin{pmatrix} \dfrac{\partial F_1}{\partial x} & \dfrac{\partial F_1}{\partial y} \\ \dfrac{\partial F_2}{\partial x} & \dfrac{\partial F_2}{\partial y} \end{pmatrix}^{-1} \right|_{\substack{x = x_k, \\ y = y_k}} \begin{pmatrix} F_1(x_k, y_k) \\ F_2(x_k, y_k) \end{pmatrix}. \tag{3}$$

для $F(x, y, n)$. Здесь

$$\begin{aligned} \frac{\partial F_1}{\partial x} &= \sum_{i=1}^{n} 3(x - x_i)^2 + (y - y_i)^2 - R_i^2; \\ \frac{\partial F_1}{\partial y} &= 2 \sum_{i=1}^{n} (x - x_i) \cdot (y - y_i) \\ \frac{\partial F_2}{\partial x} &= 2 \sum_{i=1}^{n} (x - x_i) \cdot (y - y_i); \\ \frac{\partial F_2}{\partial y} &= \sum_{i=1}^{n} (x - x_i)^2 + 3(y - y_i)^2 - R_i^2. \end{aligned}$$

Шаг 2. При поступлении новых измерений увеличиваем $n := n + 1$ и расширяем массив данных значениями x_n, y_n и R_n. Переходим на Шаг 1.

Аналогично строится алгоритм для функции $G(x, y, n)$ с матрицей производных, определяемой соотношениями

$$\begin{aligned} \frac{\partial G_1}{\partial x} &= \sum_{i=1}^{n} \left(\frac{3(x - x_i)^2 + (y - y_i)^2}{R_i^2} - 1 \right); \\ \frac{\partial G_1}{\partial y} &= 2 \sum_{i=1}^{n} \frac{(x - x_i) \cdot (y - y_i)}{R_i^2}; \\ \frac{\partial G_2}{\partial x} &= 2 \sum_{i=1}^{n} \frac{(x - x_i) \cdot (y - y_i)}{R_i^2}; \\ \frac{\partial G_2}{\partial y} &= \sum_{i=1}^{n} \left(\frac{(x - x_i)^2 + 3(y - y_i)^2 - R_i^2}{R_i^2} - 1 \right). \end{aligned}$$

Цель моделирования — показать работоспособность алгоритма, границы его применимости в рамках разработанной математической модели и поведение при отклонении от модели.

2. Реализация алгоритма

Для проведения компьютерных экспериментов представленный алгоритм реализован в предметно-ориентированной среде Scilab v5.5.0 (описание работы с этой средой смотри в [3]).

Рассмотрим разработанные функции, вошедшие в пакет моделирования. Первая функция находит точки пересечения окружностей или псевдорешения, если они не пересекаются, как это описано в [1, раздел 2.3].

```
//////////////////////////////////////////////////////////////
// Процедура поиска точек пересечения окружностей
// cp1, cp2 - массивы вида [x; y]
// n - число пересечений
// Радиусы окружностей  R1, R2 должны быть >= 0
// x1,y1 - центр первой окружности
// x2,y2 - центр первой окружности
//////////////////////////////////////////////////////////////
function [n, cp1, cp2] = SearchCrossPoints(x1,y1, R1, x2,y2, R2)
    // определим расстояние между точками, чтобы
    // определить способ пересечения
    R = sqrt((x2-x1)^2 + (y2-y1)^2);
    if or ([R>(R1+R2) R<abs(R1-R2)]) then
        // не пересекаются
        n = 0;
        // возвращаем середину отрезка
        cp1 = [(x1+x2)/2; (y1+y2)/2];
        cp2 = [(x1+x2)/2; (y1+y2)/2];
        return;
    elseif R == abs(R1-R2) then
        // одна точка пересечения - внутреннее касание
        n = 1;
        cp1 = [(x1+x2)/2; (y1+y2)/2];
        cp2 = [(x1+x2)/2; (y1+y2)/2];
        return;
    elseif R == R1+R2 then
        //одна точка пересечения
        n = 1;
        // из подобия треугольников на смещённых координатах
        cp1 = [(x2-x1)*R1/R2; (y2-y1)*R1/R2];
        cp2 = [(x2-x1)*R1/R2; (y2-y1)*R1/R2];
        return;
    end
    // основной вариант, когда точек пересечения 2.
```

```
    n = 2;
    // переносим начало координат в центр первой окружности
    // и определяем новые координаты центра второй
    x2_shift = x2 - x1;
    y2_shift = y2 - y1;
    // выполняем поворот системы координат на угол фи, так,
    // что y2_rotating = 0.
    // решая систему уравнений, получаем точки пересечения
    // в новых система координат
    x_rotating = (R1^2 - R2^2 + (x2_shift^2 + y2_shift^2))
                /(2*sqrt(x2_shift^2 + y2_shift^2));
    // Делаем обратные
    cp1_rotating = [x_rotating; sqrt(R1^2 - x_rotating^2)];
    cp2_rotating = [x_rotating; -sqrt(R1^2 - x_rotating^2)];
    // определим матрицу поворота
    a = x2_shift/sqrt(x2_shift^2 + y2_shift^2);
    b = y2_shift/sqrt(x2_shift^2 + y2_shift^2);
    MR = [a -b; b a];
    cp1_shift = MR*cp1_rotating;
    cp2_shift = MR*cp2_rotating;
    cp1 = cp1_shift + [x1; y1];
    cp2 = cp2_shift + [x1; y1];
endfunction
```

Вспомогательная функция определяет евклидово расстояние между точками плоскости в формате [x; y]

```
function distance = PointDistance(p1,p2)
    distance = sqrt(sum((p1-p2).^2));
endfunction
```

Ещё одна вспомогательная функция возвращает набор расстояний между точками массива P и искомой точкой sp

```
function R = defRad(P, sp)
    N = size(P, 2);
    R = zeros(1,N);
    for i = 1:N
        R(i) = sqrt((P(1,i) - sp(1,1))^2 + (P(2,i) - sp(2,1))^2);
    end
endfunction
```

Расставляет точки наблюдения по окружности радиусом R:

```
////////////////////////////////////////////////////////////////
// P - массив координат точек наблюдения
// N - количество точек наблюдения
// R - радиус окружности, по которой расставляются точки
```

```
////////////////////////////////////////////////////////////////
function P = defPoints(N, R)
    P = zeros(2,N);
    for i=1:N
        P(1,i)=R*cos((i-1)*2*%pi/N);
        P(2,i)=R*sin((i-1)*2*%pi/N);
    end
endfunction
```

Следующая функция определяет начальное приближение для метода Ньютона.

```
////////////////////////////////////////////////////////////////
// Поиск начального приближения по трём окружностям
////////////////////////////////////////////////////////////////
function sp2 = SearchFirstApprox(P, arr_n,  R)
    // Для начала найдем пересечение первой окружности со 2 и 3
    [n12, cp12_1, cp12_2] = SearchCrossPoints(P(1,arr_n(1)),
                     P(2,arr_n(1)), R(arr_n(1)), P(1,arr_n(2)),
                                    P(2,arr_n(2)), R(arr_n(2)));
    [n13, cp13_1, cp13_2] = SearchCrossPoints(P(1,arr_n(1)),
                      P(2,arr_n(1)), R(arr_n(1)), P(1,arr_n(3)),
                                     P(2,arr_n(3)), R(arr_n(3)));
    // определим матрицу расстояний между точками пересечения
    MD=[cp12_1 cp12_1 cp12_2 cp12_2; cp13_1 cp13_2 cp13_1 cp13_2];
    d1=PointDistance(cp12_1, cp13_1);
    d2=PointDistance(cp12_1, cp13_2);
    d3=PointDistance(cp12_2, cp13_1);
    d4=PointDistance(cp12_2, cp13_2);
    MD=[MD; d1 d2 d3 d4];
    // определим самые близкие точки и найдем первое
    // приближение к (x0 y0)
    // (MD(1,k), MD(2,k)) и (MD(3, k), MD(4, k)) - ближайшие точки
    [d,k]=min(MD(5, :));
    sp1 = [(MD(1,k) + MD(3,k))/2; (MD(2,k) + MD(4,k))/2 ];
    // ищем второе приближение (x0, y0)
    [n23, cp23_1, cp23_2] = SearchCrossPoints(P(1,arr_n(2)),
                   P(2,arr_n(2)), R(arr_n(2)), P(1,arr_n(3)),
                                  P(2,arr_n(3)), R(arr_n(3)));
    MD1 = [sp1 sp1; cp23_1 cp23_2];
    d1 = PointDistance(sp1, cp23_1);
    d2 = PointDistance(sp1, cp23_2);
    MD1 = [MD1; d1 d2];
    // определим самые близкие точки и найдём
    // первое приближение к (x0 y0)
    // (MD(1,k), MD(2,k)) и (MD(3, k), MD(4, k)) - ближайшие точки
    [d,k]=min(MD1(5, :));
    sp2 = [(MD1(1,k) + MD1(3,k))/2; (MD1(2,k) + MD1(4,k))/2];
endfunction
```

Реализация метода Ньютона для функции (1) по формулам (3)–(4)

```
function [eta, rkk] = NewtonMethodETA(N, P, spp, eps)
    // минимизируем невязку уравнений ETA
    // задаём начальное приближение
    xk = spp(1,1)
    yk = spp(2,1)
    rk=[xk+2*eps;yk+2*eps];
    rkk=[xk; yk];
    k=1;
    while  and([norm(rkk-rk,2)>eps,k<10])
        rk=rkk;
        xk=rk(1); yk=rk(2);
        F1x=0; F1y=0;
        F2x=0; F2y=0;
        F1=0;  F2=0;
        for i=1:N
          F1x=F1x+3*(xk-P(1,i))^2+(yk-P(2,i))^2-R(i)^2;
          F1y=F1y+2*(xk-P(1,i))*(yk-P(2,i));
          F2y=F2y+2*(xk-P(1,i))*(yk-P(2,i));
          F2y=F2y+(xk-P(1,i))^2+3*(yk-P(2,i))^2-R(i)^2;
          F1=F1+((xk-P(1,i))^2+(yk-P(2,i))^2-R(i)^2)*(xk-P(1,i));
          F2=F2+((xk-P(1,i))^2+(yk-P(2,i))^2-R(i)^2)*(yk-P(2,i));
        end
        Fxy=[F1x F1y; F2x F2y];
        F=[F1;F2];
        rkk=rk-inv(Fxy)*F;
        k=k+1;
    end
    eta=0;
    for i=1:1:N
      eta=eta+((rkk(1)-P(1,i))^2+(rkk(2)-P(2,i))^2-R(i)^2);
    end
endfunction
```

Реализация метода Ньютона для функции (2):

```
function [EPS, rkk] = NewtonMethodEPS(N, P, sp, eps)
// Для целевой функции EPS
// EPS-функция минимизирует ошибку измерения R
// при условии, что ошибка измерения радиуса  << R
   xk = sp(1,1)
   yk = sp(2,1)
   rk=[xk+2*eps;yk+2*eps];// лишь бы не совпадали
   rkk=[xk; yk];
   k=1;
   while and([norm(rkk-rk,2)>eps,k<10])
```

```
      xk=rk(1); yk=rk(2);
      F1x=0; F1y=0;
      F2x=0; F2y=0;
      F1=0; F2=0;
      for i=1:N
         F1x=F1x+((3*(xk-P(1,i))^2+(yk-P(2,i))^2)/R(i)^2-1);
         F1y=F1y+2*(xk-P(1,i))*(yk-P(2,i))/R(i)^2;
         F2x=F2x+2*(xk-P(1,i))*(yk-P(2,i))/R(i)^2;
         F2y=F2y+(((xk-P(1,i))^2+3*(yk-P(2,i))^2)/R(i)^2-1);
         F1=F1+(((xk-P(1,i))^2+(yk-P(2,i))^2)/R(i)^2-1)*(xk-P(1,i));
         F2=F2+(((xk-P(1,i))^2+(yk-P(2,i))^2)/R(i)^2-1)*(yk-P(2,i));
      end
      Fxy=[F1x F1y; F2x F2y];
      F=[F1;F2];
      rkk=rk-inv(Fxy)*F;
      k=k+1;
   end
   EPS=0;
   for i=1:N
     EPS=EPS+(((rkk(1)-P(1,i))^2+(rkk(2)-P(2,i))^2)/R(i)^2-1);
   end
endfunction
```

Далее представлена основная программа

```
////////////////////////////////////////////////////////////////
// Начальные данные:
// sp - искомая точка-нарушитель (x0 y0)
// N - число точек наблюдения
// (центры окружностей): p = [x1 x2 .. xN; y1 y2 ... yN]
// Sigma - величина допустимого отклонения значения радиусов
////////////////////////////////////////////////////////////////

eps=1e-5;  // погрешность для метода Ньютона
RR = 100;  // радиус разброса точек наблюдения

NN=1000;   // Число повторов каждого эксперимента
Nmax = 10; // количество точек наблюдения
Sigma = 4; // наихудшая СКО ошибки измерения
           // радиуса внутри помещений

xset("color",0); // чёрный
plot2d(0,0,-1,"031"," ",[-100,-100,100,100])
sp = [0; 0];
dR=5;
ResultX=zeros(Nmax-2,RR/dR+1);
ResultY=zeros(Nmax-2,RR/dR+1);
ResultXY=zeros(Nmax-2,RR/dR+1);
```

```
tic()
for N=3:1:Nmax
    ir=0;
    disp("-------------------------------------------");
    disp(N);
    for indR=0:dR:RR
        sp(1)=indR*cos(2*%pi*(N-3)/8);
        sp(2)=indR*sin(2*%pi*(N-3)/8);
        ir=ir+1;
        disp(ir);
        for kk=1:1:NN
            P=[]
            P = defPoints(N, RR);
            // Находим радиусы окружностей
            // с центрами в точках наблюдения
            R=[]
            R = defRad(P, sp);
            // определение ошибки измерения радиуса [0 Sigma]
            epsN = grand(1,N,'nor',0, Sigma);
            // Добавляем к радиусам ошибку
            R = R + epsN;
            // Ищем начальное приближение
            sp2 = SearchFirstApprox(P, [1 2 3], R);
            rkk=sp2;
            [eta, rkk] = NewtonMethodETA(N, P, rkk, eps)
            // вычисляем коэффициенты корреляционной матрицы
            ResultX(N-2,ir)=ResultX(N-2,ir)+1/NN*(rkk(1)-sp(1)).^2
            ResultY(N-2,ir)=ResultY(N-2,ir)+1/NN*(rkk(2)-sp(2)).^2
            ResultXY(N-2,ir)=
                 ResultXY(N-2,ir)+1/NN*(rkk(2)-sp(2))*(rkk(1)-sp(1))
       end;
    end;
end;
// Средняя дисперсия оценки
plot((0:dR:RR)',(ResultX'+ResultY')/2)
tictoctime=toc()
disp(tictoctime)
```

Работа представленных алгоритмов иллюстрируется на рисунках 1 и 2.

3. Планирование эксперимента

Будем считать, что координаты стационарных точек наблюдения известны и равномерно расположены по окружности заданного радиуса. Несанкционированная точка доступа последовательно перемещается от центра этой окружности к периферии. Ошибка измерения радиусов задаётся исследователем. Цель моделирования — построить зависимость точности оценивания от расположения точки доступа нарушителя и дисперсии ошибки измерения радиусов.

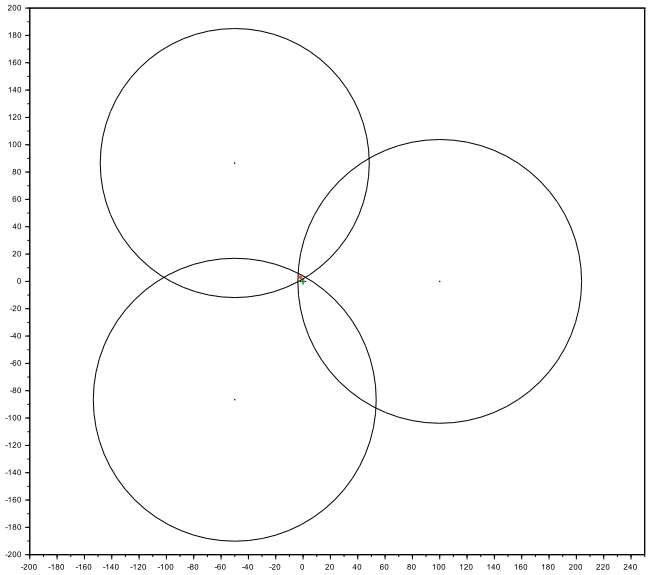

Рис. 1. Визуализация алгоритма определения координат точки беспроводного доступа по измерениям на трёх точках излучения. Зелёная окружность (окружность с крестиком) — истинное расположение (0,0), красная окружность (по центру изображения) — оценка расположения

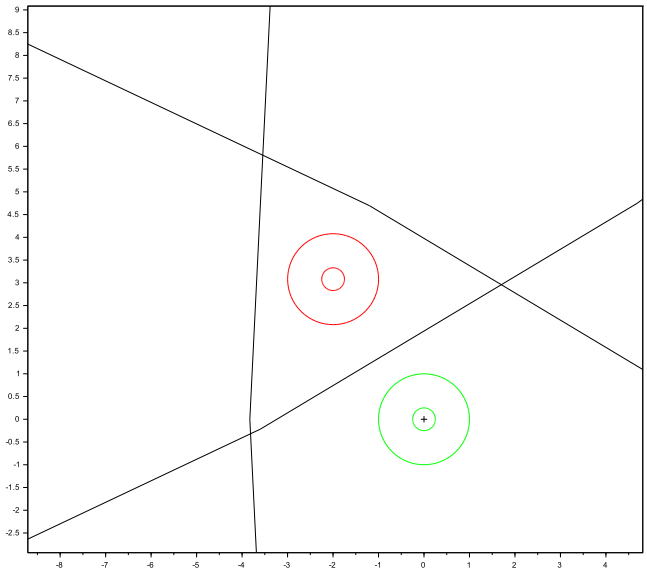

Рис. 2. Визуализация алгоритма определения координат точки беспроводного доступа по измерениям на трёх точках излучения, увеличенная версия. Зелёная окружность (расположена справа снизу) — истинное расположение (0,0), красная окружность (расположена по центру) — оценка расположения

4. Программно-аппаратное обеспечение

Эксперимент проводился на персональном компьютере со следующими характеристиками:

- Процессор: Intel-Core i7-2600K, x64.
- Тактовая частота: 3.4 ГГц.
- Ядерность: 4 физических ядра, поддержка Hyper-thrading.
- Оперативная память: 16Гб, Kingston.
- Операционная система: Windwos 7, Professianal.
- Scilab: версия 5.5.0, x64.

5. Результаты компьютерного моделирования

Проведено предварительное моделирование, которое показывает сходимость алгоритма к точке реального расположения несанкционированной точки доступа.

При проведении компьютерного эксперимента считается, что ошибка измерения представляет собой белый гауссовский шум с нулевым математическим ожиданием и дисперсией σ^2. Учитывая, что среднеквадратическое отклонение оценки радиуса по мощности сигнала в помещениях по данным экспериментов около 3-4 метров, положим для компьютерного моделирования $\sigma = 4$.

Расположение точек наблюдения осуществлялось равномерно по окружности радиусом 100 метров. Количество точек наблюдения изменялось последовательно с 3 до 10.

Несанкционированная точка доступа двигается от центра вдоль прямой к окружности расположения точек наблюдения с шагом 5 метров и направлением $\alpha = 2 * \pi/N$, где N — это число точек наблюдения.

Для оценивания дисперсии оценки координат несанкционированной точки беспроводного доступа проводилось 10000 повторений эксперимента. Результаты приведены на следующих рисунках.

Критика модельного эксперимента. Первое: ошибки оценивания радиуса могут зависеть от расстояния от точки наблюдателя до расположения нарушителя. Пока эта зависимость не исследована на практике.

Второе: предполагалось, что ошибка измерения – это гауссова случайная величина, что может оказаться неверным. Но в условиях наличия информации о первых моментах случайной величины (мат. ожидании и дисперсии) максимальная энтропия достигается на нормальном распределении. Природа как бы говорит, что реализует при данных ограничениях случайную величину максимально возможным количеством способов в случае гауссового распределения [2]. Требуется сбор статистических данных реального натурного эксперимента и выдвижение гипотез о законе распределения ошибки измерения радиуса.

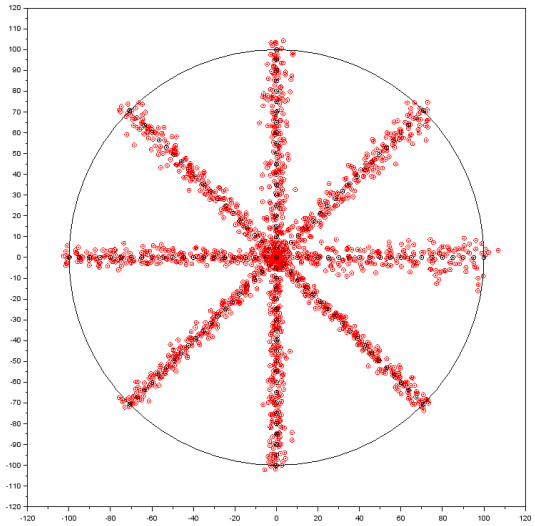

Рис. 3. Движение точки доступа от центра к окружности по направлениям: $\alpha = 0° - N = 3$, $\alpha = 45° - N = 4$, $\alpha = 90° - N = 5$, $\alpha = 135° - N = 6$, $\alpha = 180° - N = 7$, $\alpha = 225° - N = 8$, $\alpha = 270° - N = 9$, $\alpha = 305° - N = 10$, где N — число точек наблюдения. Оценки, полученные минимизацией функции $F(x, y) = ||\eta||^2$ (1), обозначены красными; истинное положение пеленгуемой точки — чёрным

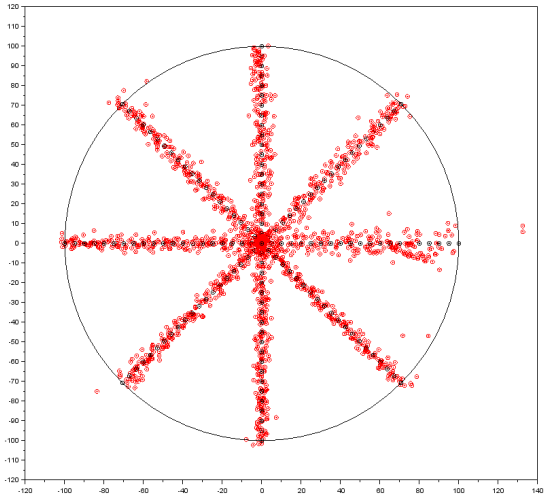

Рис. 4. Движение точки доступа от центра к окружности по направлениям: $\alpha = 0° - N = 3$, $\alpha = 45° - N = 4$, $\alpha = 90° - N = 5$, $\alpha = 135° - N = 6$, $\alpha = 180° - N = 7$, $\alpha = 225° - N = 8$, $\alpha = 270° - N = 9$, $\alpha = 305° - N = 10$, где N — число точек наблюдения. Оценки, полученные минимизацией функции $G(x, y) = ||\varepsilon||^2$ (2), обозначены красными; истинное положение пеленгуемой точки — чёрным

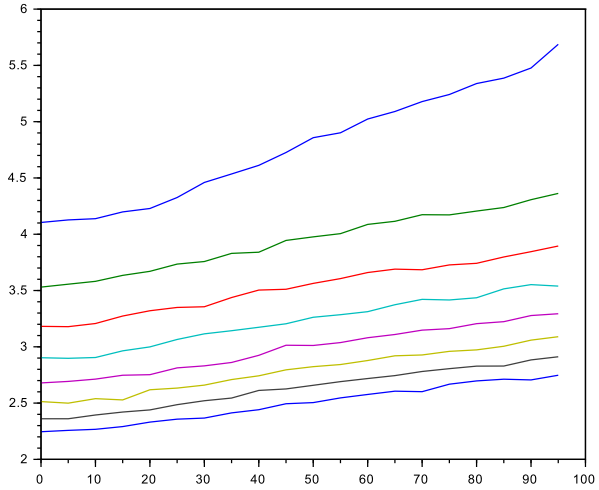

Рис. 5. Зависимость среднеквадратического отклонения от расстояния от начала координат и количества точек наблюдения для оценок, полученных минимизацией функции $F(x,y) = ||\eta||^2$ (1). $N = 3$ — синий график сверху, $N = 4$ — зелёный, $N = 5$ — красный, $N = 6$ — голубой, $N = 7$ — фиолетовый, $N = 8$ — жёлтый, $N = 9$ — чёрный, $N = 10$ — синий снизу, N — число точек наблюдения, ось X — расстояние от начала координат, ось Y — среднеквадратичное отклонение оценки от истинного расположения

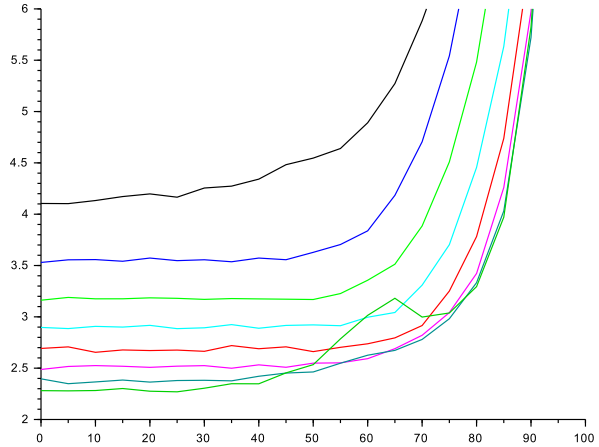

Рис. 6. Зависимость среднеквадратического отклонения от расстояния от начала координат и количества точек наблюдения для оценок, полученных минимизацией функции $G(x,y) = ||\varepsilon||^2$ (2). $N = 3$ — чёрный график, $N = 4$ — синий, $N = 5$ — ярко-зелёный, $N = 6$ — голубой, $N = 7$ — красный, $N = 8$ — фиолетовый, $N = 9$ — сине-зелёный, $N = 10$ — зелёный, N — число точек наблюдения, ось X — расстояние от начала координат, ось Y — среднеквадратичное отклонение оценки от истинного расположения

Третье: в модели никак не учитывается наличие преград и многолучевой характер распространения сигналов. Учёт этих явлений — это следующий этап исследований.

Выводы

1. Метод, основанный на минимизации невязки уравнений, более устойчив к геометрии размещения точек наблюдения. По видимому, этот метод будет более устойчив и к другим несоответствиям модели и действительности.

2. Метод, основанный на минимизации ошибки измерения радиуса, на расстояния более 30 – 40 метров даёт более точные оценки, дисперсия ошибки при этом практически не возрастает. Но этот метод гораздо более чувствителен и имеет существенно большие ошибки, по сравнению с предыдущим, когда пеленгуемая точка и точки наблюдения сближаются. Связано это с формой целевой функции, у которой в знаменателе стоит расстояние от цели до точки наблюдения, которое, в свою очередь, стремится к нулю при сближении пеленгуемой точки и точки наблюдения. К сожалению, ограничение в 30 – 40 метров, ранее упомянутое, фактически говорит о неприменимости данного подхода на практике. Неожиданным оказался результат связанный с потерей сходимости метода Ньютона в случае, когда точек наблюдения 10. Причину этой потери сходимости предстоит выяснить в дальнейших исследованиях.

3. Более стабильно ведёт себя первый метод — метод минимизации невязки уравнений. Но дальнейшее изучение улучшения точности оценки должно, видимо, вестись в рамках второго метода — метода минимизации ошибки измерения радиуса.

В дальнейших исследованиях необходимо:

- в рамках модели оценить влияние ошибок измерения координат наблюдателей на ошибку измерения положения точки доступа;
- уточнить модель реального шума измерений и его влияния на оценку координат, проведя и обработав натурные эксперименты;
- по результатам моделирования уточнить модель измерений и провести повторные компьютерные эксперименты;
- реализовать клиентское приложение для мобильных платформ (смартфонов), обеспечивающее сбор данных о мощностях сигналов точек доступа и координатах наблюдателя;
- реализовать серверное приложение, обеспечивающее получение данных от наблюдателей и производящее расчёт искомого расположения точки доступа;
- провести натурные эксперименты по пеленгу несанкционированных точек;

- разработать модель, учитывающую естественные и искусственные преграды.

Литература

1. Вишнякова О.А., Лавров Д.Н., Лаврова С.Ю. Математическая модель обнаружения точки беспроводного доступа по измерениям мощности излучения разнесёнными наблюдателями // Математические структуры и моделирование. 2013. №2(28). С. 49–59.
2. Джейнс Э.Т. О логическом обосновании метода максимальной энтропии // ТИИЭР. 1982. Т. 70, № 9. С. 33–51.
3. Тропин И.С., Михайлова О.И., Михайлов А.В. Численные и технические расчёты в среде Scilab (ПО для решения задач численных и технических вычислений) : Учебное пособие. М. , 2008. 64 с.

COMPUTER MODELING OF ESTIMATING OF WIRELESS ACCESS POINT COORDINATES BY MEASURING THE RECEIVED SIGNAL POWER

D.N. Lavrov[1]
Ph.D. (Eng.), Associate Professor, e-mail: lavrov@omsu.ru
O.A. Vishnyakova[2]
Software Engineer, e-mail: olga@infotekorg.ru
E.I. Dudyak[3]
Software Engineer, e-mail: gggorec@gmail.com
S.U. Lavrova[1]
Master of Computer Science, e-mail: Sveta.Lavrova@gmail.com

[1]Omsk State University n.a. F.M. Dostoevskiy
[2]Aspekt, Ltd.
[3]PJSC “Omskiy Nauchno Issledovatelskiy Institut Priborostroeniya” (PJSC “ONIIP”)

Abstract. This paper presents the code and the results of computer modeling determining the coordinates of a wireless access point. In the previous work we built a mathematical model of the distributed meter. This work is devoted to the verification of operability and quality of the proposed algorithms. Two objective functions were subjected to comparative analysis. According to the results of the computer simulation, it was concluded that the method based on the objective function minimizing the residual of the system of equations is preferable.

Keywords: access point, wireless network, radiolocation, Newton-Raphson method, computer modeling.