Санкт-Петербургский государственный университет

**Кафедра технологии программирования**

**Магомаева Аида Руслановна**

**Выпускная квалификационная работа бакалавра**

**Совместное применение эвристических**

**вероятностных алгоритмов и метода заряженных шариков для нахождения ближайшей к нулю точки множества**

Направление 010300

«Фундаментальная информатика и информационные технологии»

Научный руководитель,

кандидат физ.-мат. наук,

доцент
Аббасов Меджид Эльхан оглы

**Содержание**

[Введение 3](#_Toc480743226)

[Постановка задачи 5](#_Toc480743227)

[Глава 1. Обзор литературы 6](#_Toc480743228)

[Глава 2. Теоретическая часть 9](#_Toc480743229)

[2.1. Эвристические алгоритмы 9](#_Toc480743230)

[2.2. Метод заряженных шариков 11](#_Toc480743231)

[Глава 3. Практическая часть 17](#_Toc480743232)

[3.1. Итоговый алгоритм 17](#_Toc480743233)

[3.2. Результаты работы алгоритма 17](#_Toc480743234)

[Выводы 22](#_Toc480743235)

[Заключение 23](#_Toc480743236)

[Список литературы 24](#_Toc480743237)

[Приложение 25](#_Toc480743238)

# Введение

В общем смысле совокупность методов, ориентированных на нахождение наибольших и наименьших значений величин, называют теорией оптимизации. Основным разделом теории оптимизации является нелинейное программирование (НЛП), задача которого охватывает мириады форм и возникает в физических, естественных, математических, экономических науках и в инженерной деятельности.

Так, в экономике — это основная экономическая задача, связанная с распределением ограниченных ресурсов. Также решение задачи НЛП используется в управлении государством, когда нужно минимизировать затраты при условии, что благосостояние государства должно находиться выше некоторого минимального значения.

В стандартной задаче НЛП требуется минимизировать целевую нелинейную функцию нескольких переменных при заданном допустимом множестве :

 (1)

Для их решения применяются итеративные методы, идеей которых является движение от заданной начальной точки вдоль генерируемой последовательности точек , приближающейся к некоторому «дачи оптимальному» решению данной задачи. Переход от к называется итерацией. Обычно она строится по схеме

где вектор — направление в точке , а число — длина шага. Итеративные методы отличаются друг от друга способом вычисления .

Правила остановки итеративного процесса задаются различными способами с заданной точностью решения исходной задачи:

Эффективность процесса оптимизации, с одной стороны, зависит от умения использовать результаты таких наук, как теория алгоритмов, функциональный анализ, математический анализ и линейная алгебра.

С другой стороны, решения оптимизационной задачи в некоторых случаях требуют больших затрат временных и вычислительных ресурсов. Из-за этого такие задачи в первую очередь направлены на реализацию на вычислительной технике.

Одними из наиболее исследованных являются задачи выпуклого программирования. Этот раздел оптимизации занимается минимизацией выпуклых функций на выпуклых множествах. И это не случайно, ведь свойство выпуклости чрезвычайно важно не только в теоретическом плане (единственность решения, выполнение условий регулярности и т.д.), так и в практическом (существует большое количество методов разработанных для выпуклых задач и учитывающих их специфику). Однако, как правило, задачи, возникающие на практике, являются невыпуклыми.

В данной работе рассматривается задача нахождения ближайшей к нулю точки невыпуклого множества. Рассматривается алгоритм поиска глобального решения поставленной задачи. Он основан на совместном применении метода заряженных шариков, изначально предлагавшегося для поиска ортогональной проекции точки на выпуклое множество, а также эвристических вероятностных алгоритмов. Последние используются для поиска исходной точки в окрестности глобального решения, из которой далее запускается метод заряженных шариков.

Отметим, что, вообще говоря, задача получения начальной точки является самостоятельной важной задачей, которая особенно важна в алгоритмах глобальной оптимизации.

Рассмотренная ниже задача имеет множество практических приложений и возникает, например, при моделировании освещения в компьютерной графике, робототехнике, астрономии и т.д.

# Постановка задачи

Пусть — произвольная точка множества , где — непрерывно дифференцируемая функция.

Требуется найти ближайшую к нулю точку множества :

 (2)

Следует отметить, что задачу (2) можно расширить на случай нахождения минимального расстояния между множеством и любой точкой , при помощи преобразования (сдвига) координат, приняв за .

Очевидно, что — замкнутое множество. Для любого минимум будет достигаться на множестве , где — замкнутый шар радиуса с центром в начале координат. Это множество замкнуто как пересечение двух замкнутых множеств, и ограничено как пересечение множеств, одно из которых ограничено. Исходя из вышесказанного, по теореме Вейерштрасса существует такая точка , на которой достигается минимальное значение целевой функции Отметим, что в отличие от [1] мы не требуем выпуклости Х.

Ожидается, что решение задачи (2) будет найдено для любой заданной точности .

# Глава 1. Обзор литературы

При написании данной работы были использованы научная и учебно-методическая литература. Основными источниками, раскрывающими теоретические основы нелинейного программирования, являлись работы Зангвилла У. «Нелинейное программирование. Единый подход.» [2], Васильева Ф.П. «Численные методы решения экстремальных задач» [3] и Моисеева Н.М. «Методы оптимизации» [4]. В данных источниках подробно рассмотрены основные понятия теории оптимизации, ее место в общей научной картине мира и методы решения задач нелинейного программирования.

При решении поставленной задачи были использованы алгоритмы, описанные в статьях М. Э. Аббасова «Метод заряженных шариков» [1] и «Эвристические вероятностные алгоритмы ортогонального проектирования точки на множество» [5].

Ниже представлены некоторые существующие методы решения общей задачи нелинейного программирования (1), дающие возможность находить глобальные экстремумы в случаях, когда задачи обладают соответствующими свойствами выпуклости и вогнутости. Для невыпуклых задач численные методы оптимизации не гарантируют поиск глобального решения, так как найденные решения могут существенно зависеть от начального приближения. Поиск глобального решения в общем случае чрезвычайно сложен. Для этого можно, в частности, использовать дополнительные методы нахождения начальной точки в некоторой окрестности глобальных экстремумов.

**1.1. Метод возможных направлений**

Данный метод подробно изложен в [1]. Суть его заключается в движении точки по некоторому подходящему направлению, не приводящему к нарушению огранчиений задачи и уменьшающему значения целевой функции.

Он основан на следующем замечании: если - выпуклые дифференцируемые функции и для направления выполнены условия

где — множество индексов активных ограничений в точке , для которых выполнено условие, а — некоторое положительное число, то существует .

Тогда направление движения определяется из решения следующей задачи линейного программирования

где , , где M – предельное число итераций.

Таким образом, метод возможных направлений сводит исходную задачу нелинейного программирования к последовательности задач линейного программирования, где целевая функция является линейной.

**1.2. Методы штрафных функций**

Методы штрафных функций изложены в [2]. Идея этих методов заключается сведении задачи (1) к эквивалентной задаче без ограничений:

где - штрафная функция и параметр — коэффициент штрафа.

Общий вид функции :

где , .

В зависимости от вида различают метод внешних штрафных функций и метод внутренних штрафных (или барьерных) функций.

**Метод внешних штрафных функций**

В данном методе штрафная функция задается таким образом, чтобы на границе и внутри допустимого множества равнялась нулю, а при нарушениях ограничений «штрафовала» функцию , увеличивая ее значение. Поиск можно начинать из произвольной точки.

Примером штрафной функции может быть

здесь .

**Метод внутренних штрафных функций или барьерных функций**

Главным недостатком предыдущего метода является то, что полученные приближения не принадлежат множеству , так как решение задачи аппроксимируется снаружи. В методе внутренних штрафных функций оно аппроксимируется изнутри. Значения соответствующей функции штрафа возрастают при приближении к границе допустимой области. Поиск минимума функции начинается с внутренней точки множества , а траектория движения всегда будет принадлежать эту множеству.

Примером барьерной функции может быть

.

# Глава 2. Теоретическая часть

Для решения задачи (2) были выбраны алгоритмы, которые предложил Аббасов М.Э. в работах [3] и [4]. Каждый из них рассматривался в качестве отдельных алгоритмов, объединив которые, можно расширить область их применения на невыпуклый случай. В данной работе будет исследовано их совместное применение для решения задачи (2).

## 2.1. Эвристические алгоритмы

Алгоритмы, предложенные в [3], являются первичным этапом нахождения ближайшей к нулю точки множества. Как уже было сказано выше, трудность исследуемой задачи (2) заключается в том, что при сложной структуре допустимого множества успех существующих алгоритмов ее решения зависит от начальной точки, из которой они запускаются. Эвристические алгоритмы позволяют находить хоть и грубое, но вполне достаточное приближение к глобальному минимуму, обеспечивающее корректную работу алгоритмов по решению задачи (2).

**Базовый алгоритм**

Пусть Возьмем некоторое , по достижении которого алгоритм завершает свою работу, и любую точку , такую, что , где — открытый шар радиуса r с центром в начале координат, и пусть j=1.

Опишем алгоритм. Предположим, что мы находимся на k-ой итерации.

1. С помощью многомерного распределения на генерируем случайный вектор , и проверяем условие принадлежности множеству: ;

2. При выполнении условия присваиваем и переходим к шагу 1;

3. При невыполнении условия:

1. если , то и переходим к шагу 1;
2. если , то , процесс окончен.

Иллюстрация работы алгоритма представлена на рисунке 2.1.1.

Рис 2.1.1. Иллюстрация работы базового алгоритма

**Алгоритм с половинным делением**

Базовый алгоритм можно усилить, введя идею половинного деления. Пусть — открытый шар радиуса *R* с центром в начале координат без центральной части радиуса *r*. Возьмем любую точку ,

Опишем алгоритм. Предположим, что мы находимся на k-ой итерации.

1. С помощью многомерного распределения на генерируем случайный вектор , и проверяем условие принадлежности множеству: ;

2. При выполнении условия присваиваем и переходим к шагу 4;

3. При невыполнении условия:

1. если , то и переходим к шагу 1;
2. если , то и переходим к шагу 4.

4. Проверяем критерий остановки :

1. при его выполнении присваиваем , процесс окончен;
2. при невыполнении критерия присваиваем и переходим к шагу 1.­­

Иллюстрация работы алгоритма представлена на рисунке 2.1.2.

a) b)

Рис 2.1.2. Случайные точки генерируются в закрашенной области.

Случаи: a) . b) .

**2.2. Метод заряженных шариков**

Метод заряженных шариков предложен в работе [4]. Идея данного алгоритма заключается в переходе от исходной задачи оптимизации к механической аналогии. Безусловным преимуществом такого метода является интуитивное понимание и уверенность в его сходимости, вытекающие из законов механики. Такая система с течением времени стремится к равновесному положению, совпадающему с решением исходной задачи, и делает возможным построение новых эффективных итерационных алгоритмов. Строятся они путем составления дифференциальных уравнений движения с последующим переходом к разностной схеме их решения. Класс получаемых таких образом задач называется методами установления, подразумевая установление равновесия в нестационарной механической системе. Подробно методы установления описаны в [6].

Метод заряженных шариков продолжает идеи хорошо известного метода тяжелого шарика, наиболее популярного представителя класса методов установлений. В нем для нахождения минимума функции необходимо поместить на поверхность тяжелый шарик, который может двигаться только по данной поверхности . Очевидно, что с течением времени шарик окажется в положении, соответствующему локальному минимуму , то есть остановится в решении исходной задачи.

Иллюстрация работы метода тяжелого шарика показана на рисунке 2.2.1.

Рис 2.2.1. Иллюстрация работы метода тяжелого шарика.

**Реализация алгоритма**

Для решения задачи (2) поместим в начало координат и зафиксируем отрицательный заряд . Возьмем произвольную точку и поместим в нее положительно заряженный шарик массой и зарядом . Под действием силы Кулона шарик будет двигаться в направлении начала координат по прямой, их соединяющей, пока он не столкнется с границей множества . В предположении, что сила трения отсутствует, шарик продолжит двигаться по внутренней стороне поверхности, заданной функцией , достигнет положения равновесия, которое совпадает с решением задачи (2), и начнет колебаться около этого положения. Действительно, в любой другой точке касательная составляющая силы Кулона не равна нулю и направлена к положению равновесия. Для того, чтобы процесс сошелся с решению задачи (2), можно ввести силу сопротивления, пропорциональную скорости и противоположно ей. В таком случае колебанию станут затухающими.

Обозначим координаты шарика , где время. Сила Кулона равна

где — электрическая постоянная. Сила нормальной реакции перпендикулярна поверхности в точке , равна нормальной составляющей кулоновской силы и направлена внутрь множества:

Силу сопротивления введем формулой

где — коэффициент сопротивления.

По второму закону Ньютона в инерциальных системах отсчёта ускорение, приобретаемое материальной точкой, прямо пропорционально вызывающей его силе, совпадает с ней по направлению и обратно пропорционально массе материальной точки:

Подставив значения сил в правой части, получим уравнение:

 (3)

Перейдем от (3) к системе первого порядка с помощью введения ‑мерного вектора фиктивных переменных :

 (4)

где

а и — параметры, зависящие от Решать систему (4) мы будем методом ломанных Эйлера. Пусть и выберем некоторое — длину шага.

Тогда

 (5)

Так как при составлении уравнений не учитывается центробежная сила, то с ростом *k* точка будет все дальше отдаляться от истинной траектории (рис. 2.2.2).

Рис 2.2.2. С ростом *k* траектория решения выводит точку за границу множества. *x(\*)* — положение равновесия.

Для возвращения точки истинную траекторию введем специальную процедуру, проектирующую эту точку на границу множества . Предположим, что, совершая итерацию по (5) из точки , мы попадаем в точку . При небольшом шаге точка лежит в малой окрестности границы множества , и можно считать, что прямая, проходящая параллельно через точку , пересекает поверхность множества под прямым углом. Вследствие чего, в качестве будем брать точку этого пересечения, как проекцию на границу множества .

Итак,

где — некоторый скаляр, определяемый из условия

Разложим эту функцию в окрестности и отбросим все слагаемые выше первого порядка, учитывая близость к границе множества :

Отсюда следует, что

 (6)

Итого, алгоритм (5) перепишется в виде

 (7)

Иллюстрация работы алгоритма (7) представлена на рисунке 2.2.3.

a) b)

c) d)

Рис 2.2.3. Иллюстрация работы алгоритма (7): a) на точка движется к x\* — положению равновесия, b) точка «пролетает» x\*, останавливается и меняет направление движения, c) точка снова «пролетает» x\*, но останавливается ближе к положению равновесия и снова меняет направление движения, d) на k+3-шаге точка еще ближе останавливается к x\*.

**Критерий сходимости**

Сила Кулона уравновешивается силой нормальной реакции в положении равновесия, и векторы и в этом положении являются коллинеарными. Поэтому в качестве критерия сходимости можно взять условие

 (8)

где — некоторое малое число. Критерий основан на том, что равно касательной составляющей силы Кулона в точке . Очевидно, норма равна нулю тогда и только тогда, когда вектора и коллинеарны.

Также можно получить коллинеарность векторов и и из необходимых условий экстремума для поставленной задачи условной оптимизации. Используя метод множителей Лагранжа, запишем функцию Лагранжа:

где — некоторый скаляр. Приравнивая к нулю частные производные функции Лагранжа по , получим условие, которому должно удовлетворять решение задачи (2):

что означает коллинеарность векторов и .

# Глава 3. Практическая часть

**3.1. Итоговый алгоритм**

Метод заряженных шариков предполагает решение выпуклых задач, в противном случае алгоритм может остановиться в точке локального экстремума. Чтобы расширить данный алгоритм на невыпуклый случай, метод заряженных шариков необходимо запускать из точки, находящейся в окрестности глобального экстремума, где множество локально выпукло. Для нахождения такой точки мы будем использовать описанные выше эвристические алгоритмы.

Однако проблема заключается в том, что эвристические алгоритмы позволяют найти точку в окрестности глобального экстремума, в то время как для корректной работы метода заряженных шариков начальная точка должна находиться в на границе множества . Для преодоления этой проблемы мы будем использовать процедуру корректировки (6).

Пусть — точка, полученная при запуске одного их эвристических алгоритмов из п.2.1. Применив для нее процедуру корректировки (6) получим точку из окрестности глобального решения, находящуюся в малой окрестности границы множества . Из полученного таким образом приближения запустим метод заряженных шариков и получим итоговый алгоритм решения задачи (2):

 (9)

В качестве критерия сходимости будем использовать условие (8).

**3.2. Результаты работы алгоритма**

Покажем, как итоговый алгоритм работает на примерах. Задача каждый раз решается 20 раз, выводятся средние значения отклонения от точного решения, средний критерий остановки алгоритма и среднее время работы алгоритмов в секундах. Все вычисления проводились при . Алгоритм реализован в математическом пакете MATLAB R2012b.

Стоит отметить, что параметры , и существенно влияют на скорость работы алгоритма. Их выбор — отдельная задача.

**Пример 1**

Пусть в нужно найти расстояние от начала координат до овала Декарта (рис. 3.2.1):



Рис.3.2.1. Область, которую образует множество из примера 1.

Таблица 3.2.1. Результаты совместного применения эвристических алгоритмов и алгоритма (9) на примере 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Эвристический алгоритм |  |  | сходимости | истинного решения | работыалгоритма (сек) |
| Базовый | 100 | 5 | 5.4071×10-7 | 1.7136×10-6 | 4.802 |
| 100 | 20 | 4.3404×10-7 | 1.7154×10-6 | 1.268 |
| 100 | 50 | 5.4012×10-7 | 1.7129×10-6 | 0.665 |
| 100 | 150 | 9.9972×10-7 | 1.7121×10-6 | 3.860 |
| С половинным делением | 100 | 5 | 3.9142×10-7 | 1.7124×10-6 | 5.848 |
| 100 | 20 | 3.8944×10-7 | 1.7131×10-6 | 2.550 |
| 100 | 50 | 7.2579×10-7 | 1.7148×10-6 | 2.530 |
| 100 | 150 | 9.9965×10-7 | 1.7137×10-6 | 12.591 |

В дальнейшем в результатах вычислений будут показаны наиболее оптимальные зафиксированные при численных расчетах значения и .

**Пример 2**

Пусть в нужно найти расстояние от начала координат до множества (рис. 3.2.2):



Рис.3.2.2. Область, которую образует множество из примера 2.

В таблице 3.2.2. приведены результаты решения примера 2.

Таблица 3.2.2. Результаты совместного применения эвристических алгоритмов и алгоритма (9) на примере

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Эвристический алгоритм |  |  | сходимости | истинного решения | работыалгоритма (сек) |
| Базовый | 100 | 50 | 5.4744×10-7 | 1.8098×10-6 | 0.656 |
| С половинным делением | 100 | 50 | 7.7695×10-7 | 1.8046×10-6 | 1.903 |

**Пример 3**

Пусть в нужно найти расстояние от начала координат до множества (рис. 3.2.3):



Рис.3.2.3. Область, которую образует множество из примера 3.

В таблице 3.2.3. приведены результаты решения примера 3.

Таблица 3.2.3. Результаты совместного применения эвристических алгоритмов и алгоритма (9) на примере

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Эвристический алгоритм |  |  | сходимости | истинного решения | работыалгоритма (сек) |
| Базовый | 100 | 15 | 5.7853×10-7 | 1.2672×10-6 | 1.543 |
| С половинным делением | 100 | 15 | 5.9301×10-7 | 1.2682×10-6 | 2.702 |

**Пример 4**

Пусть в нужно найти расстояние от начала координат до множества (рис. 3.2.4):



Рис.3.2.4. Область, которую образует множество из примера 4.

В таблице 3.2.4. приведены результаты решения примера 4.

Таблица 3.2.4. Результаты совместного применения эвристических алгоритмов и алгоритма (9) на примере

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Эвристический алгоритм |  |  | сходимости | истинного решения | работыалгоритма (сек) |
| Базовый | 100 | 15 | 4.7367×10-7 | 1.7652×10-6 | 1.681 |
| С половинным делением | 100 | 15 | 4.0075×10-7 | 1.7632×10-6 | 14.445 |

Стоит отметить, что в данном примере встроенная функция «fmincon» системы MATLAB, позволяющая решить задачу (2), выдала ответ с точностью решения 10-7, в то время как точность решения итогового алгоритма оказалась 10-17.

**Выводы**

В данной дипломной работе была предложена схема решения задачи (2) на основе совместного применения существующих эвристических алгоритмов и метода заряженных шариков. Проведены численные эксперименты и показана работоспособность и эффективность указанной схемы. Комбинация рассмотренных методов позволила получить решение с высокой точностью и в примере 4 показала себя эффективнее, чем одна из существующих функций в пакете вычислений MATLAB, которая является наиболее используемой для решения соответствующей задачи.

Преимуществом итогового алгоритма по сравнению с другими методами является независимость от выбора начальной точки. Отметим, что чем сложнее допустимое множество, тем дольше время работы алгоритма, так как появляется необходимость в увеличении количества итераций для нахождения точки вблизи глобального экстремума. Кроме того, открытой является задача выбора оптимальных значений параметров , и .

**Заключение**

В работе была рассмотрена задача нахождения ближайшей к нулю точки невыпуклого множества с гладкой границей и предложен алгоритм для ее решения. Данный алгоритм был построен на основе недавно предложенных эвристических алгоритмах и методе заряженных шариков. Проведенные вычислительные эксперименты показали возможность практического применения предложенного подхода в различных задачах, и в некоторых случаях доказали его эффективность по сравнении с уже существующими алгоритмами, используемыми для решения аналогичных задач.

**Список литературы**

1. Аббасов М.Э. «Метод заряженных шариков» // Семинар по конструктивному гладкому анализу и недифференцируемой оптимизации «CNSA & NDO», 2015.
2. Зангвилл У.И. Нелинейное программирование. Единый подход. – М.: Наука. 2009.
3. Васильев Ф.П. Численные методы решения экстремальных задач. - М.: Наука, 1980.
4. Моисеев Н.М. Методы оптимизации. - М.: Наука, 1978.
5. Аббасов М.Э. «Эвристические вероятностные алгоритмы ортогонального проектирования точки на множество» // Семинар по конструктивному гладкому анализу и недифференцируемой оптимизации «CNSA & NDO», 2015.
6. Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. Численные методы. М.: Наука, 1987.

**Приложение**

Ниже представлены фрагменты программы для реализации алгоритма (9).

**Итоговый алгоритм (9):**

N=20;

krit\_all=zeros(1,N);

rast\_all=zeros(1,N);

meaning=zeros(2,N);

for i=1:N

d=0.0001; p1=100; p2=16; k=0; eps=0.000001;

X=try\_nachalnoe;

baz=X;

Z=[0, 0];

krit=1;

while krit>eps

 k=k+1;

 df(1)=dfx(X(1), X(2));

 df(2)=dfy(X(1), X(2)) ;

 Xv=X+d\*Z;

 dfv(1)=dfx(Xv(1), Xv(2));

 dfv(2)=dfy(Xv(1), Xv(2)) ;

 psi=-p1\*X./(norm(X)^3)+p1\*(X\*df')\*df/((norm(X)^3)\*(norm(df)^2))-p2\*Z;

 Z=Z+d\*psi;

 X=Xv-dfv\*funct(Xv(1), Xv(2))/(norm(dfv)^2);

 krit=norm(X(2)/X(1)-dfv(2)/dfv(1));

end

end

**Получение начальной точки:**

function baz=try\_nachalnoe

d=0.0001; p1=100; p2=100; k=1;

baz=test\_heuristic\_algorithms\_bazoviy;

%baz=test\_heuristic\_algorithms\_polov\_delenie;

Z(k, :)=[0, 0];

j=1; eps\_b=0.000001;

while abs(funct(baz(1), baz(2)))>eps\_b

 j=j+1;

 k=k+1;

 df(1)=dfx(baz(1), baz(2));

 df(2)=dfy(baz(1), baz(2)) ;

 Xv=baz+d\*Z(k-1, :);

 dfv(1)=dfx(Xv(1), Xv(2)) ;

 dfv(2)=dfy(Xv(1), Xv(2)) ;

 psi=-p1\*baz./(norm(baz)^3)+p1\*(baz\*df')\*df/((norm(baz)^3)\*(norm(df)^2))-

 p2\*Z(k-1, :);

 Z(k, :)=Z(k-1, :)+d\*psi;

 baz=Xv-dfv\*funct(Xv(1), Xv(2))/(norm(dfv)^2);

end

**Функция запуска эвристического базового алгоритма:**

function Xok=test\_heuristic\_algorithms\_bazoviy

X(1, :)=[-2,-20];

m=10000;

j=1; X(1, :)=[2,2];

k=1;

while j<=m

 rad=sqrt(X(k, 1).^2+X(k, 2).^2);

 Y(j, :)=ravn\_rasp(rad);

 if funct(Y(j, 1), Y(j, 2)) <=0

 X(k+1, :)=Y(j, :);

 k=k+1;

 else

 if j<m

 j=j+1;

 else if j==m

 Xok=X(k, :);

 j=j+1;

 end

 end

 end

end

**Функция запуска эвристического алгоритма с половинным делением:**

function Xok=test\_heuristic\_algorithms\_polov\_delenie

X=[2,2];

r=0; R=norm(X);

m=10000; j=1; eps=0.00001;

while j<=m

 Y=ravn\_rasp\_polov(r, (R+r)/2);

 if funct(Y(1), Y(2)) <=0

 R=norm(Y);

 X=Y;

 if norm(R-r)<=eps

 Xok=X;

 j=m+1;

 end

 else

 if j<m

 j=j+1;

 else if j==m

 r=(R+r)/2;

 if norm(R-r)<=eps

 Xok=X;

 j=m+1;

 end

 end

 end

 end

end