

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования «Казанский национальный исследовательский
технический университет им. А.Н. Туполева-КАИ»
(КНИТУ-КАИ)

Институт компьютерных технологий и защиты информации
(наименование института (факультета), филиала)
Кафедра прикладной математики и информатики
(наименование кафедры)

01.03.02 «Прикладная математика и информатика» профиль «Исследование операций и системный анализ»
(шифр и наименование направления подготовки (специальности))

К защите допустить

Зав. каф. ПМИ

_____ Зайдуллин С.С.

«__» _____ 2019 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

на тему: Реализация численных методов решения негладких экстремальных
задач

ОБУЧАЮЩИЙСЯ Бирюков А. М.
(инициалы, фамилия) (личная подпись)

РУКОВОДИТЕЛЬ д.т.н., проф. Заботин В. И.
(ученая степень, звание, инициалы, фамилия) (личная подпись)

Казань 2019

MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE RUSSIAN FEDERATION
Federal State Educational Institution of Higher Education
«Kazan National Research Technical University named after A. N. Tupolev-KAI»
(KNRTU-KAI)

Institute of Computer Technologies and Information Protection

(Institute name)

Department of Applied Mathematics and Informatics

(department name)

01.03.02 «Applied Mathematics and Informatics» field «Operations Research and System Analysis»
(reference code and name of area of training (speciality))

Admitted to defense

**Head of Applied Mathematics and
Informatics Department**

«__» _____ 2019

FINAL QUALIFYING WORK

On the theme of Implementation of numerical methods for solving nonsmooth
extremal problems

STUDENT Biryukov Aleksey

(initials, surname)

(personal signature)

ACADEMIC ADVISOR D. Sc., Full Professor Zabotin Vladislav

(initials, surname)

(personal signature)

Kazan 2019

**Федеральное государственное бюджетное образовательное учреждение
высшего образования «Казанский национальный исследовательский технический
университет им. А.Н. Туполева-КАИ»
(КНИТУ-КАИ)**

Институт (факультет), филиал _____ ИКТЗИ _____
Кафедра _____ ПМИ _____
Направление/специальность _____ 01.03.02 Прикладная математика и информатика _____

УТВЕРЖДАЮ
Заведующий кафедрой

« _____ » _____ 20__ г.

ЗАДАНИЕ
выпускной квалификационной работы

Бирюков Алексей Михайлович
(фамилия, имя, отчество)

1 Тема выпускной квалификационной работы

Реализация численных методов решения негладких экстремальных задач

утверждена приказом по университету от " _____ " _____ 20__ г. № _____

2 Срок сдачи обучающимся законченной выпускной квалификационной работы 30.05.19

3 Исходные данные к выпускной квалификационной работе _____

1). Арутюнова, Н.К. Модификация метода Евтушенко поиска глобального минимума для случая непрерывной на компакте функции / Н.К. Арутюнова // Вестник КГТУ им. А.Н. Туполева. – 2013. - № 2, вып. 2. – С. 154-157.

2). Vanderbei, R.J. Extension of Piyavskii's Algorithm to Continuous Global Optimization / R.J. Vanderbei // Journal of Global Optimization. – 1999. – Vol. 14. – P. 205-216.

3). Заботин, В.И. Алгоритм вычисления минимальной оценки ε -постоянной Липшица непрерывной функции / В.И. Заботин, П.А. Чернышевский // Вестник КГТУ им. А.Н. Туполева. – 2018. - № 2, вып. 2. – С. 127-132.

4). Васильев, Ф. П. Численные методы решения экстремальных задач (2-е издание) / Ф. П. Васильев. – Москва: «НАУКА», 1988. – 551 с.

4 Содержание расчётно-пояснительной записки (перечень подлежащих разработке вопросов и исходные данные к ним):

4.1 Аннотация

Введение

Глава 1. Алгоритм равномерного перебора отыскания глобального минимума функции одной переменной непрерывной(ϵ -Липшицевой) на отрезке

а) Описание алгоритма

б) Обоснование алгоритма

Глава 2. Алгоритм равномерного перебора отыскания глобального минимума функции одной переменной непрерывной(ϵ -Липшицевой) на двумерном брус

а) Описание алгоритма

б) Обоснование алгоритма

Глава 3. Расчёт численных примеров

Глава 4. Программная реализация алгоритмов и инструкция пользователя

Заключение

Список используемых источников

Приложения

Приложение 1. Листинг программы

Приложение 2. Презентация

5 Перечень графического материала (с точным указанием обязательных чертежей):

Слайд 1. Титульный слайд

Слайд 2. Введение

Слайд 3. Постановка задачи для одномерного случая

Слайд 4. Алгоритм минимизации непрерывной функции одной переменной

Слайд 5. Обоснование алгоритма

Слайд 6. Пример

Слайд 7. Постановка задачи для функции двух переменных на брус

Слайд 8. Алгоритм минимизации непрерывной функции двух переменных

Слайд 9. Обоснование алгоритма

Слайд 10. Пример

Слайд 11. Расчёт тестовых примеров

Слайд 12. Структурная схема алгоритма для функции двух переменных на брус

Слайд 13. Заключение

6 Консультанты по ВКР (с указанием относящихся к ним разделов):

Раздел	Консультант (фамилия и инициалы)	Подпись, дата	
		Задание выдал	Задание принял
Основной раздел	Заботин В. И.	01.10.18	20.05.19

7 Дата выдачи задания 01.10.18

Руководитель ВКР _____
(подпись)

Заботин В. И. _____
(фамилия и инициалы)

Задание к исполнению принял _____
(подпись)

П р и м е ч а н и е.

1 Задание прилагается к законченному выпускной квалификационной работы и вместе с пояснительной запиской представляется в ГЭК.

2 Перед началом выполнения выпускной квалификационной работы обучающийся разрабатывает календарный план работы с указанием очередности выполнения отдельных этапов, согласовывает его с руководителем выпускной квалификационной работы.

Календарный план

№ п/п	Наименование этапов выпускной квалификационной работы	Срок выполнения этапов выпускной квалификационной работы	Примечание
1	Получение задания у руководителя	01.10.18	
2	Изучение источников	02.10.18-26.10.18	
3	Изучение существующих алгоритмов	20.10.18-16.11.18	
4	Разработка алгоритма минимизации негладкой функции одной переменной	18.11.18-30.11.18	
5	Обоснование алгоритма	29.11.18-14.12.18	
6	Расчёт численных примеров	15.12.18-29.12.18	
7	Разработка алгоритма минимизации негладкой функции двух переменных	30.12.18-25.02.19	
8	Обоснование алгоритма	26.02.19-20.03.19	
9	Расчёт численных примеров	21.03.19-18.04.19	
10	Написание пояснительной записки и разработка презентации	19.04.19-20.05.19	

Обучающийся _____

Руководитель _____

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	11
ГЛАВА 1. АЛГОРИТМ РАВНОМЕРНОГО ПЕРЕБОРА ОТЫСКАНИЯ ГЛОБАЛЬНОГО МИНИМУМА ФУНКЦИИ ОДНОЙ ПЕРЕМЕННОЙ НЕПРЕРЫВНОЙ (ϵ -ЛИПШИЦЕВОЙ) НА ОТРЕЗКЕ	14
Описание алгоритма и его обоснование	14
Замечание к алгоритму	16
Численный пример	17
ГЛАВА 2. СЛУЧАЙ ФУНКЦИИ ДВУХ ПЕРЕМЕННЫХ	20
Описание алгоритма и его обоснование	20
Замечание к алгоритму	22
Численный пример	23
ГЛАВА 3. РАСЧЁТ ЧИСЛЕННЫХ ПРИМЕРОВ	24
ГЛАВА 4. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМОВ	28
Обоснование выбора языка и среды программирования	28
Проектирование программного обеспечения	29
Реализация алгоритма равномерного перебора отыскания глобального минимума непрерывной (ϵ -липшицевой) на отрезке функции одной переменной	30
Реализация алгоритма для случая функции двух переменных	32
ГЛАВА 5. ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА	36
Основные понятия и принципы тестирования программного продукта	36
Тестирование, применённое к разрабатываемому программному обеспечению	38
Реализация тестовых примеров	39
РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ	41
Руководство по эксплуатации	41
Вспомогательная программа (Визуализатор)	43
ЗАКЛЮЧЕНИЕ	46
CONCLUSION	47
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	48
ПРИЛОЖЕНИЕ 1. Письмо с подтверждением принятия публикации	50

ПРИЛОЖЕНИЕ 2. Сертификат участника конференции	51
ПРИЛОЖЕНИЕ 3. Листинг программного продукта	52
ПРИЛОЖЕНИЕ 4. Листинг вспомогательного программного продукта (Визуализатор).....	56
ПРИЛОЖЕНИЕ 5. Листинг разработанных модульных тестов	59
ПРИЛОЖЕНИЕ 6. Презентация	62

АННОТАЦИЯ

В работе предлагаются и обосновываются легко реализуемые численные методы приближённого поиска глобального минимума непрерывной на отрезке функции и непрерывной на бруске функции двух переменных, на которые не накладываются никакие дополнительные условия типа липшицевости. Единственным требованием к функции является её непрерывность на выпуклом компакте (отрезке или прямоугольнике). Приводятся результаты численных экспериментов. Методы могут быть полезны при решении задач негладкого математического программирования.

ABSTRACT

The paper proposes and proves an easily implemented numerical methods for the approximate search for a global minimum of a continuous function on an interval and of a continuous function of two variables on a rectangle without any additional constraints such as conditions of Lipschitz type. The only requirement for the function is its continuity on a convex set (interval or rectangle). The results of numerical experiments are given. Methods can be useful in solving problems of non-smooth mathematical programming.

ВВЕДЕНИЕ

Уже достаточно давно исследователями ставится задача нахождения глобального минимума функции одной или нескольких переменных. Ясно, что методы отыскания локального минимума не решают поставленную задачу, поскольку не дают уверенности в том, что найдены все локальные минимумы функции. Нужны какие-либо глобальные свойства функции на множестве ограничений. Одним из таких свойств является выпуклость функции на выпуклом множестве. Однако, это достаточно узкий класс функций. Другой, более широкий класс, это класс липшицевых функций на выпуклом множестве. В настоящее время известно достаточно много численных методов отыскания глобального минимума липшицевой функции. По-видимому, первым из таких методов является хорошо известный ныне метод Пиявского (метод ломаных). В след за указанным методом появились методы Евтушенко, Стронгина и другие.

Разумеется, что эти методы не решают задачу глобальной оптимизации для произвольной непрерывной функции, которая необязательно является липшицевой. Такой, например, является функция $\sqrt{|x|}$ на отрезке $[-1; 1]$, очевидно, имеющая глобальный минимум в точке $x = 0$.

Как выяснилось в работе [1], любая непрерывная на выпуклом компакте функция обладает некоторым глобальным свойством, обобщающим свойство липшицевости, которое может позволить создавать приближённые методы отыскания глобального минимума любой непрерывной на выпуклом компакте функции.

В предлагаемой работе строятся и обосновываются два приближённых алгоритма минимизации функции непрерывной на отрезке и функции непрерывной на замкнутом прямоугольнике.

Поясним вышесказанное.

В работе [1] было введено понятие ε -липшицевости:

Пусть функция f определена на выпуклом компактном множестве A из нормированного пространства R^n . Она называется ε -липшицевой, если выполняется следующее условие:

$$\forall \varepsilon > 0 \exists L_\varepsilon \forall x, y \in A: |f(x) - f(y)| \leq L_\varepsilon \|x - y\| + \varepsilon \quad (1)$$

Там же было доказано, что выполнение условия (1) **необходимо и достаточно** для непрерывности функции f на A . Оказалось, что свойство ε -липшицевости функции можно с успехом применять к созданию методов оптимизации негладких(может быть существенно негладких) функций. Так, в указанной работе [1], была предложена модификация метода Пиявского (метода ломаных) поиска приближённого минимума ε -липшицевой функции. Таким образом, открывается возможность минимизации непрерывных на компакте функций без добавления каких-либо дополнительных условий (кроме непрерывности) на минимизируемую функцию. Ясно, что в силу известной теоремы Вейерштрасса этот минимум существует, осталось его найти.

Следует заметить, что из определения (1) не следует положительности постоянной L_ε , которую будем называть ε -константой Липшица.

Для дальнейших выкладок для нас будет существенной её строгая положительность. В [2] было показано, что неравенство $L_\varepsilon > 0$ при заданном $\varepsilon > 0$ выполняется, если найдутся $x, y \in A$ такие, что

$$|f(x) - f(y)| > \varepsilon \quad (2)$$

Условие (2) было названо согласованностью функции f и $\varepsilon > 0$. Легко понять, что это условие не сильно сужает класс рассматриваемых функций, и мы будем всюду в дальнейшем полагать это условие выполненным.

Свойство ε -липшицевости вызвало появление ряда работ [3, 4, 5, 6], в каждой из которых приходилось решать вспомогательную задачу минимизации ε -липшицевой функции. При этом предполагалось, что значение оценки L_ε было известно или известна функциональная зависимость L_ε от ε . Отыскание указанных зависимостей является непростой задачей. Можно сказать, что непростой задачей является уже задача нахождения постоянной Липшица для

липшицевых функций, задача же нахождения постоянной L_ε является задачей ещё более сложной. В последнее время появилась работа [2], в которой предложен один из возможных алгоритмов приближённого отыскания постоянной L_ε по заданному $\varepsilon > 0$. В данной работе считается, что эта задача решена и нам известна минимальная оценка L_ε .

Одним из известных простых в реализации методов минимизации липшицевой функции является метод перебора по равномерной сетке. В предлагаемой работе строится и обосновывается обобщение упомянутого метода на случай ε -липшицевой функции.

В работе принята сквозная нумерация формул.

ГЛАВА 1. АЛГОРИТМ РАВНОМЕРНОГО ПЕРЕБОРА ОТЫСКАНИЯ ГЛОБАЛЬНОГО МИНИМУМА ФУНКЦИИ ОДНОЙ ПЕРЕМЕННОЙ НЕПРЕРЫВНОЙ (Е-ЛИПШИЦЕВОЙ) НА ОТРЕЗКЕ

Описание алгоритма и его обоснование

Здесь мы сформулируем и обоснуем алгоритм минимизации непрерывной функции на отрезке.

Пусть f непрерывна на отрезке $[a; b]$, L_ε её ε -постоянная Липшица, которая считается известной. Предполагается выполненным условие согласованности f и ε (2): найдутся $x, y \in A$ такие, что

$$|f(x) - f(y)| > \varepsilon$$

Введём параметры алгоритма:

- ε – параметр, выбираемый из условия (1)
- $\varepsilon^* > 0$ – погрешность, с которой отыскивается приближённое значение минимума функции

Всюду предполагается выполнение неравенства $\varepsilon^* > \varepsilon$.

Обозначим:

- f^* - наименьшее значение функции на отрезке $[a; b]$ (которое, очевидно, существует);
- положим $h = \frac{b-a}{n}$, где n -количество отрезков, на которые разбивается отрезок $[a; b]$.

Потребуем выполнение неравенства

$$h \leq \frac{\varepsilon^* - \varepsilon}{L_\varepsilon}$$

отсюда $n \geq \frac{L_\varepsilon(b-a)}{\varepsilon^* - \varepsilon}$. Полученное условие позволяет найти количество пробных точек – узлов равномерной сетки, по которой ведётся перебор.

Построим последовательность пробных точек отрезка по следующему правилу:

$$x_{i+1} = x_i + h, i = 0, 1, \dots, n - 1$$

Нашей задачей будет показать, что

$$|\min\{f(x_i), i = 0, 1, \dots, n - 1\} - f^*| \leq \varepsilon^*$$

или, что то же самое

$$\min\{f(x_i), i = 0, 1, \dots, n - 1\} - f^* \leq \varepsilon^*.$$

Очевидно, для этого будет достаточно показать, что среди членов последовательности найдётся точка x_k , для которой выполняется условие:

$$f(x_k) - f^* \leq \varepsilon^* \quad (3)$$

Обозначим $\Delta_i = [x_i; x_{i+1}]$ и пусть $x \in \Delta_i$. Тогда из неравенства (1) следует

$$-L_\varepsilon |x - x_i| - \varepsilon \leq f(x) - f(x_i)$$

или, что то же самое

$$f(x) \geq f(x_i) - L_\varepsilon |x - x_i| - \varepsilon \quad (4)$$

Но поскольку $|x - x_i| \leq h$, то выполняется следующее неравенство

$$-L_\varepsilon |x - x_i| \geq -L_\varepsilon h$$

Тогда из неравенства (4) следует:

$$f(x) \geq f(x_i) - L_\varepsilon h - \varepsilon = f(x_i) - L_\varepsilon \frac{\varepsilon^* - \varepsilon}{L_\varepsilon} - \varepsilon = f(x_i) - \varepsilon^* \quad (5)$$

Пусть x^* - точка глобального минимума функции f на отрезке $[a; b]$.

Очевидно, найдётся отрезок Δ_k такой, что $x^* \in \Delta_k$.

Тогда, из неравенства (5) следует:

$$f(x_k) \leq f(x^*) + \varepsilon^*$$

Что и требовалось доказать.

Замечание к алгоритму

Использование алгоритма предполагает априорное знание ε -константы Липшица для заданного ε .

Как видно из цитируемой литературы - лишь для некоторых элементарных функций удалось построить функциональную зависимость L_ε от ε . Последняя зависимость, из известных автору, получена в работе [6] для функции $\arcsin(x)$, что, как следует из работы, было сопряжено с трудностями. Однако, если дополнить приведённый алгоритм предварительным нахождением минимальной оценки ε -константы Липшица с помощью алгоритма, предложенного в [2], то указанная трудность снимается (правда ценой увеличения времени нахождения глобального минимума).

Численный пример

В качестве тестовой задачи приближённого нахождения глобального минимума функции была рассмотрена функция вида [5]:

$$f(x) = \min \{ \sqrt{|x - a_1|} + b_1, \sqrt{|x - a_2|} + b_2, \sqrt{|x - a_3|} + b_3 \}$$

для которой в [5] была получена оценка

$$L(\varepsilon) = \frac{1}{4\varepsilon}$$

Для двух различных наборов постоянных a_i, b_i приведём результаты численного эксперимента.

Таблица 1. Результаты работы алгоритма

Отрезок	ε^*	ε	$L(\varepsilon)$	h	x_{min}	F_n	n	$t, \text{мс}$
Набор 1. $b_1 = b_2 = b_3 = 0; a_1 = -4, a_2 = -1, a_3 = 3$								
[-5; 5]	0.01	0.005	50	0.0001	-0.999999	0.000001	99999	4
		0.001	250	0.000036	-1.000004	0.002000	277777	11
	0.001	0.0005	500	0.000001	-3.999999	0.000012	9999999	398
		0.0001	2500	0.0000004	-1.000000	0.000202	27777777	1101
[-10; 10]	0.01	0.005	50	0.000100	2.999999	0.000003	200000	9
		0.001	250	0.000036	-1.000000	0.000007	555555	35
	0.001	0.0005	500	0.000001	2.999999	0.000026	20000000	751
		0.0001	2500	0.0000004	-1.000000	0.000081	55555555	2059
[-15; 15]	0.01	0.005	50	0.000100	2.999999	0.000004	300000	12
		0.001	250	0.000036	2.999999	0.000009	833333	32
	0.001	0.0005	500	0.000001	2.999999	0.000066	30000000	1148
		0.0001	2500	0.0000004	2.999999	0.000110	83333333	3059

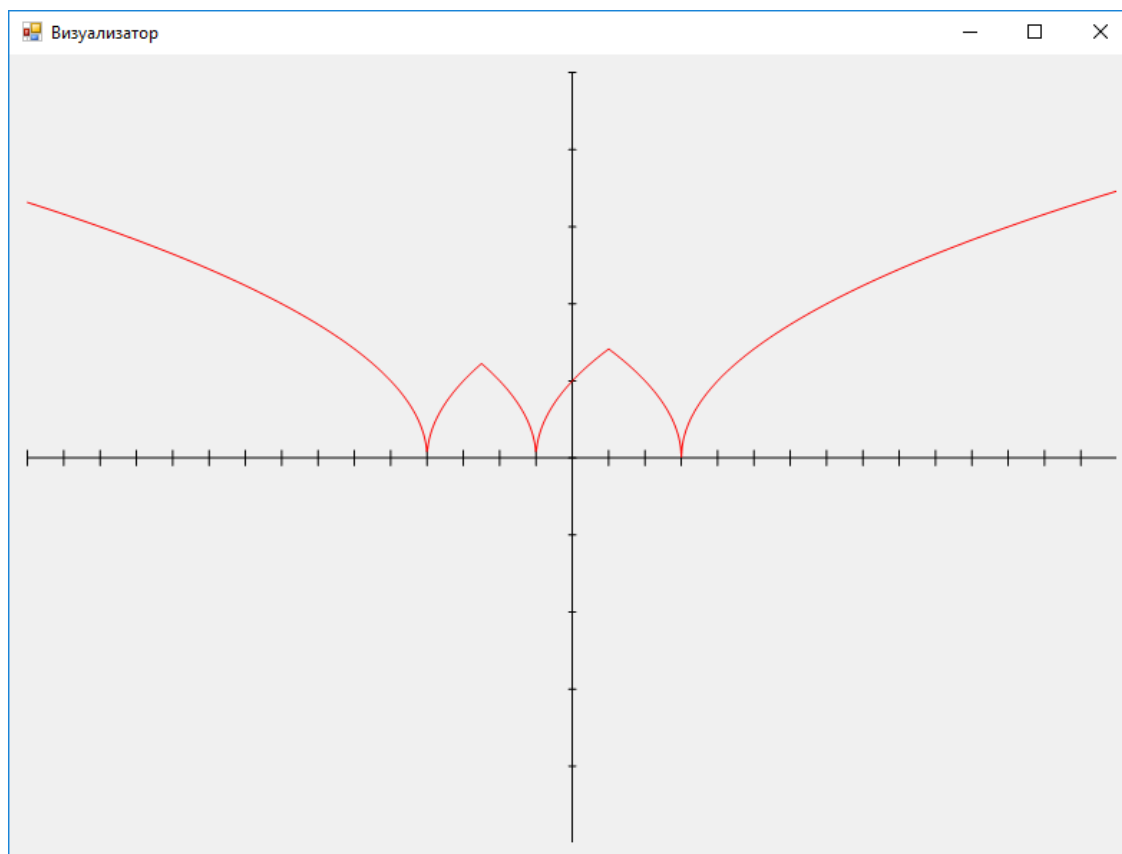


Рисунок 1 График функции с первым набором параметров

Отрезок	ε^*	ε	$L(\varepsilon)$	h	x_{min}	F_n	n	$t, \text{мс}$
Набор 2. $b_1 = -1, b_2 = -1.005, b_3 = 0.5; a_1 = -4, a_2 = -1, a_3 = 3$								
[-5; 5]	0.01	0.005	50	0.000100	-0.999999	-1.004999	99999	3
		0.001	250	0.000036	-1.000004	-1.003000	277777	9
	0.001	0.0005	500	0.000001	-0.999999	-1.004982	9999999	367
		0.0001	2500	0.0000004	-1.000000	-1.004798	27777777	997
[-10; 10]	0.01	0.005	50	0.000100	-1.000000	-1.004997	200000	7
		0.001	250	0.000036	-1.000000	-1.004993	555555	23
	0.001	0.0005	500	0.000001	-1.000000	-1.004973	20000000	711
		0.0001	2500	0.0000004	-1.000000	-1.004919	55555555	1971
[-15; 15]	0.01	0.005	50	0.000100	-1.000000	-1.004995	300000	10
		0.001	250	0.000036	-0.999996	-1.003000	833333	43
	0.001	0.0005	500	0.000001	-1.000000	-1.004933	30000000	1056
		0.0001	2500	0.0000004	-1.000000	-1.004834	83333333	2871

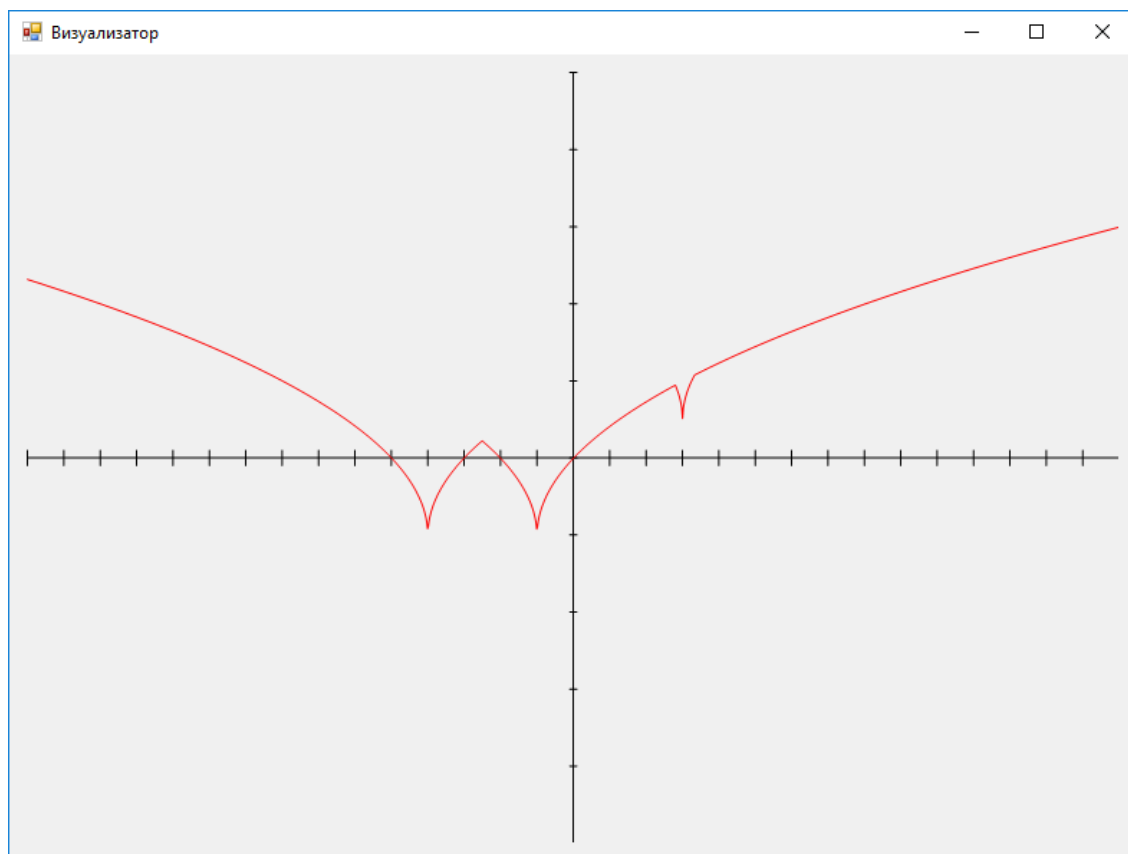


Рисунок 2 График функции со вторым набором параметров

Сравнивая с результатами, полученными в [5], мы видим, что при более простой реализации алгоритма требуется больше пробных точек, однако время, необходимое на выполнение алгоритма, значительно уменьшается.

Другие примеры работы алгоритма будут показаны ниже в главе 3.

ГЛАВА 2. СЛУЧАЙ ФУНКЦИИ ДВУХ ПЕРЕМЕННЫХ

Описание алгоритма и его обоснование

Далее рассмотрим случай применения метода равномерного перебора поиска глобального минимума ε -липшицевой функции двух переменных.

Дана функция $f(x, y)$ непрерывная на $[a; b] \times [d; c]$, L_ε её ε -постоянная Липшица, которая считается известной. Предполагается выполненным условие согласованности f и ε (2): найдутся $x, y \in A$ такие, что

$$|f(x_1, y_1) - f(x_2, y_2)| > \varepsilon$$

Прямоугольник, на котором будет производиться поиск минимума, разделим сеткой размера $n \times m$. Введём следующие обозначения:

- ε – параметр, выбираемый из условия (1)
- $\varepsilon^* > 0$ – погрешность, с которой отыскивается приближённое значение минимума функции
- $h_y = \frac{c-d}{m}$ – шаг алгоритма по оси Oy;
- $h_x = \frac{b-a}{n}$ – шаг алгоритма по оси Ox;
- $f^* = (x^*; y^*)$ – наименьшее значение функции на области $[a; b] \times [d; c]$ (которое, очевидно, существует);
- $(x_i; y_j) \in \Delta_{ij}$ – узел сетки, являющийся нижней левой точкой ячейки Δ_{ij} , где $i = \overline{0, n-1}, j = \overline{0, m-1}$, на которые разбивается прямоугольник $[a; b] \times [d; c]$.

Определим значение m . Для этого потребуем выполнения неравенства

$$h_y \leq \frac{\varepsilon^* - \varepsilon}{L_\varepsilon}$$

Тогда, очевидно

$$\frac{c-d}{m} \leq \frac{\varepsilon^* - \varepsilon}{L_\varepsilon}$$

И, преобразовав это неравенство, получаем нижнюю границу величины шага по оси Oy :

$$m \geq \frac{(c - d)L_\varepsilon}{\varepsilon^* - \varepsilon}$$

Аналогичным образом получаем оценку n числа разбиений по оси Ox :

$$n \geq \frac{(b - a)L_\varepsilon}{\varepsilon^* - \varepsilon}$$

Нашей задачей будет доказать истинность следующего неравенства:

$$|\min\{f(x_i, y_j), i = 0, 1, \dots, n - 1, j = 0, 1, \dots, m - 1\} - f^*| \leq \varepsilon^*$$

Найдётся Δ_{ij} такой, что $(x^*; y^*) \in \Delta_{ij}$

$$f(x_i, y_j) - f(x^*; y^*) \leq L_\varepsilon \|(x_i, y_j) - (x^*; y^*)\| + \varepsilon \quad (6)$$

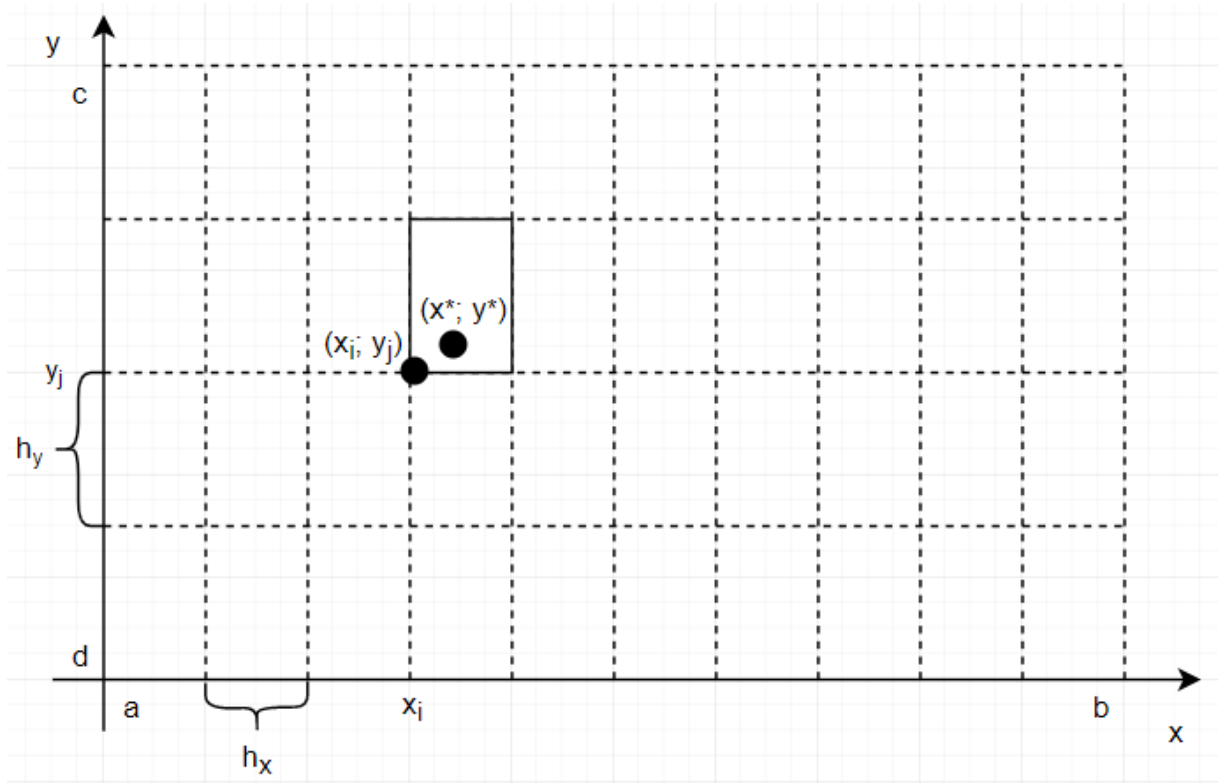


Рисунок 3 Геометрическое пояснение

Или, что то же самое

$$f(x^*; y^*) \geq f(x_i, y_j) - L_\varepsilon \|(x_i, y_j) - (x^*; y^*)\| - \varepsilon$$

В качестве нормы удобно будет выбрать покомпонентную норму

$$\|(a; b)\| = \max \{|a|; |b|\}$$

Тогда:

$$f(x^*; y^*) \geq f(x_i, y_j) - L_\varepsilon \max\{|x_i - x^*|; |y_j - y^*|\} - \varepsilon$$

Так как соседние узлы сетки не могут располагаться друг от друга больше, чем на величину шага, то

$$f(x^*; y^*) \geq f(x_i, y_j) - L_\varepsilon \max\{h_x; h_y\} - \varepsilon$$

А так как и h_x , и h_y ограничены сверху одной величиной $\frac{\varepsilon^* - \varepsilon}{L_\varepsilon}$, можно записать

$$f(x^*; y^*) \geq f(x_i, y_j) - L_\varepsilon \frac{\varepsilon^* - \varepsilon}{L_\varepsilon} - \varepsilon = f(x_i, y_j) - \varepsilon^*$$

Откуда получаем требуемое неравенство

$$f(x_i, y_j) \leq f(x^*; y^*) + \varepsilon^*$$

Что и требовалось доказать.

Замечание к алгоритму

Строго говоря, ε -константы Липшица по осям Ox и Oy могут различаться, но можно взять большую из них и использовать как универсальную

$$L_\varepsilon = \max\{L_{\varepsilon_x}; L_{\varepsilon_y}\}$$

И тогда величина шага по обеим осям станет одинаковой

$$h_x = h_y \leq \frac{\varepsilon^* - \varepsilon}{L_\varepsilon}$$

Впрочем, небольшое усложнение алгоритма возможно и для различающихся констант L_{ε_x} и L_{ε_y} .

Численный пример

В качестве тестовой задачи приближённого нахождения глобального минимума функции была рассмотрена функция вида

$$f(x, y) = |x| + \sqrt{|\sin y|}$$

для которой в [1] была получена оценка

$$L(\varepsilon) = \frac{1}{4\varepsilon} + 1$$

Таблица 2. Результаты работы алгоритма

Брус	ε^*	ε	$L(\varepsilon)$	h	x_{min}	y_{min}	F_n	n	m	$t, \text{мс}$
$[-1; 1]$ \times $[-\frac{\pi}{2}; \frac{\pi}{2}]$	0.05	0.01	26	0.001538	0.000000	-0.000027	0.005210	1300	2043	117
		0.001	251	0.000195	-0.000088	-0.000063	0.008044	10245	16093	7191
	0.01	0.005	51	0.000098	0.000000	-0.000012	0.003472	20400	32045	28118
		0.001	251	0.000036	0.000004	0.000008	0.002904	55778	87616	221548
	0.05	0.01	26	0.001538	0.000000	0.000000	0.000000	650	2043	65
		0.001	251	0.000195	-0.000044	0.000000	0.000044	5123	16093	3627
$[-0.5; 0.5]$ \times $[0; \pi]$	0.05	0.01	26	0.001538	0.000000	0.000000	0.000000	650	2043	65
		0.001	251	0.000195	-0.000044	0.000000	0.000044	5123	16093	3627
	0.01	0.005	51	0.000098	0.000000	0.000000	0.000000	10200	32045	14249
		0.001	251	0.000036	-0.000016	0.000000	0.000016	27889	87616	106840
	0.05	0.01	26	0.001538	0.000000	0.000000	0.000000	650	2043	65
		0.001	251	0.000195	-0.000044	0.000000	0.000044	5123	16093	3627

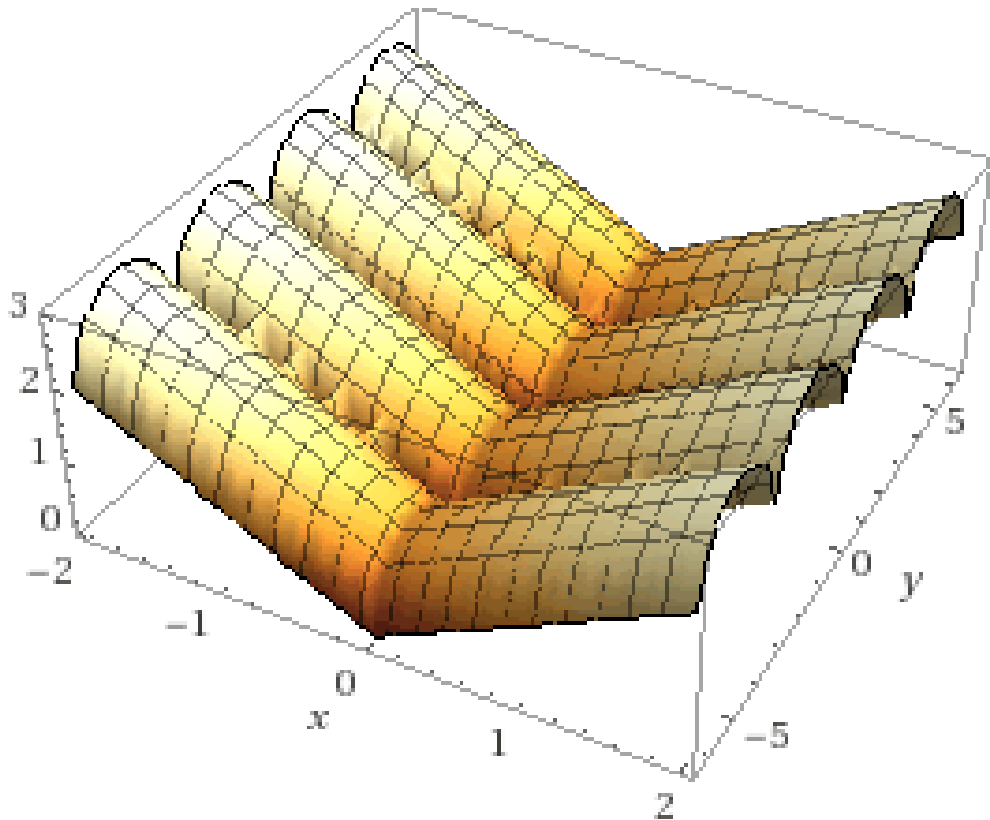


Рисунок 4 График рассматриваемой функции

ГЛАВА 3. РАСЧЁТ ЧИСЛЕННЫХ ПРИМЕРОВ

Помимо приведённых численных примеров в предыдущих главах, был проведён ряд численных экспериментов над разработанными алгоритмами с другими функциями, некоторые из которых будут приведены ниже.

$$1. f(x) = \sqrt{|x|} + |\sin x|$$

на интервале $[-2\pi; 3\pi]$, известна её ε -константа Липшица

$$L(\varepsilon) = \frac{1}{4\varepsilon} + 1$$

Таблица 3. Результаты работы алгоритма

Отрезок	ε^*	ε	$L(\varepsilon)$	h	x_{min}	F_n	n	$t, \text{мс}$
$[-2\pi; 3\pi]$	0.01	0.005	51	0.000098	0.000044	0.006647	2796	0
		0.001	251	0.000036	-0.000013	0.0036	7645	0
	0.001	0.0005	501	0.000001	0.000000	0.000636	274703	12
		0.0001	2501	0.0000004	0.000000	0.00031	761848	35

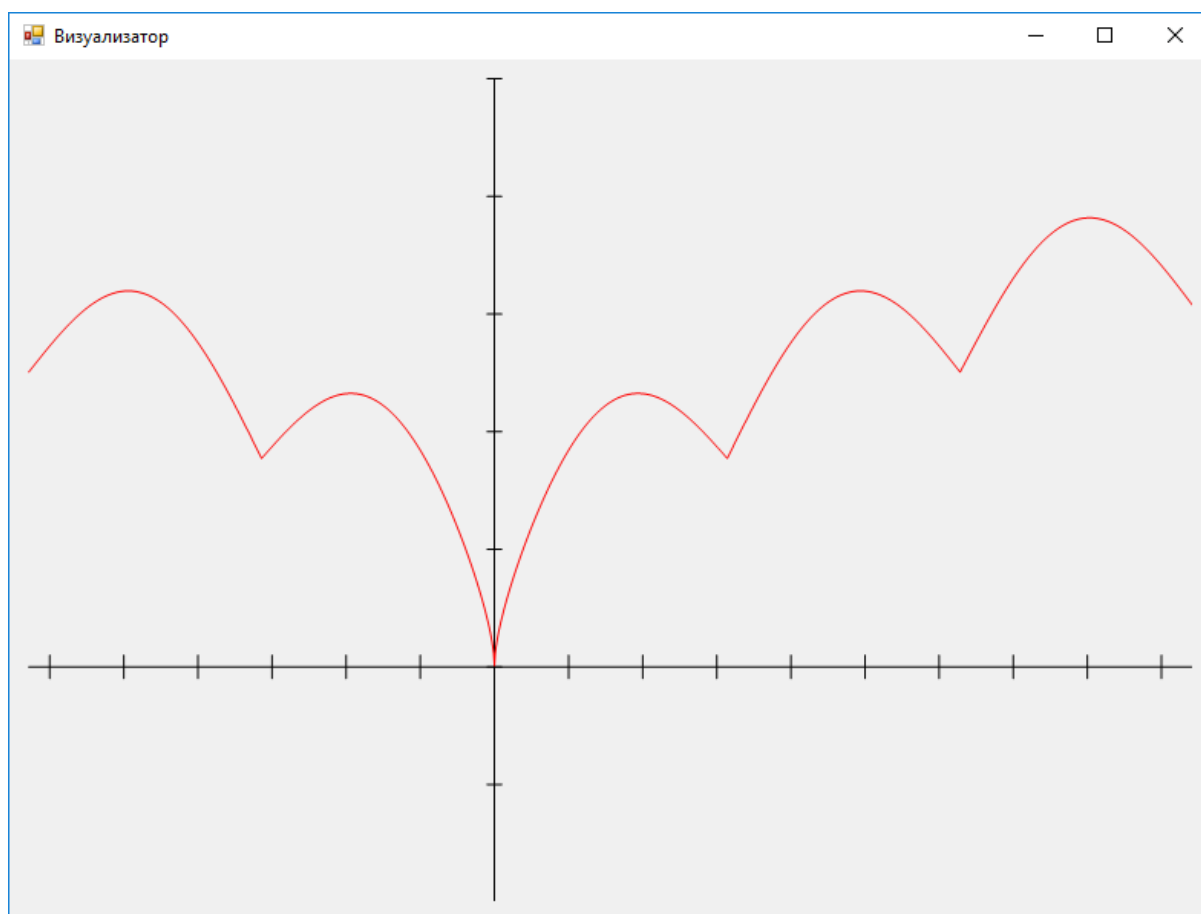


Рисунок 5 График исследуемой на минимум функции

$$2. f(x) = |x| + \sqrt{|\sin x|}$$

на интервале $[-2\pi; 3\pi]$, известна её ε -константа Липшица

$$L(\varepsilon) = \frac{1}{4\varepsilon} + 1$$

Таблица 4. Результаты работы алгоритма

Отрезок	ε^*	ε	$L(\varepsilon)$	h	x_{min}	F_n	n	$t, \text{мс}$
$[-2\pi; 3\pi]$	0.01	0.005	51	0.000098	0.000044	0.006647	2796	0
		0.001	251	0.000036	-0.000013	0.0036	7645	0
	0.001	0.0005	501	0.000001	0.000000	0.000636	274704	14
		0.0001	2501	0.0000004	0.000000	0.00031	761848	32

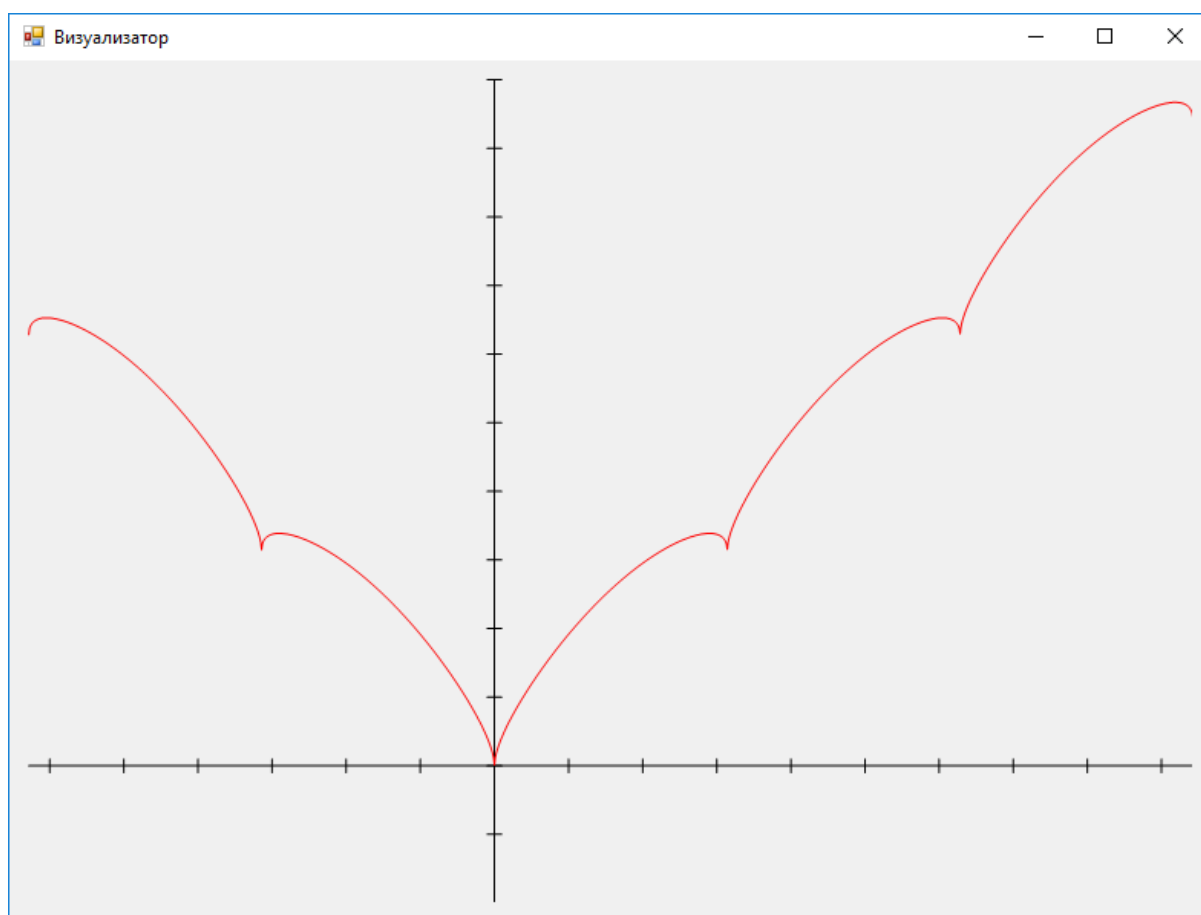


Рисунок 6 График исследуемой на минимум функции

$$3. f(x, y) = |y| + \sqrt{|\sin x|}$$

на брус $[-\frac{\pi}{2}; \frac{\pi}{2}] \times [-1; 1]$, известна её ε -константа Липшица

$$L(\varepsilon) = \frac{1}{4\varepsilon} + 1$$

Таблица 5. Результаты работы алгоритма

Брус	ε^*	ε	$L(\varepsilon)$	h	x_{min}	y_{min}	F_n	n	m	$t, \text{мс}$
$[-\frac{\pi}{2}; \frac{\pi}{2}]$ \times $[-1; 1]$	0.05	0.01	26	0.001538	-0.000027	0.000000	0.005210	2043	1300	143
		0.001	251	0.000195	-0.000063	-0.000088	0.008044	16093	10245	7166
	0.01	0.005	51	0.000098	-0.000012	0.000000	0.003472	32045	20400	28302
		0.001	251	0.000036	0.000008	0.000004	0.002904	87616	55778	212089

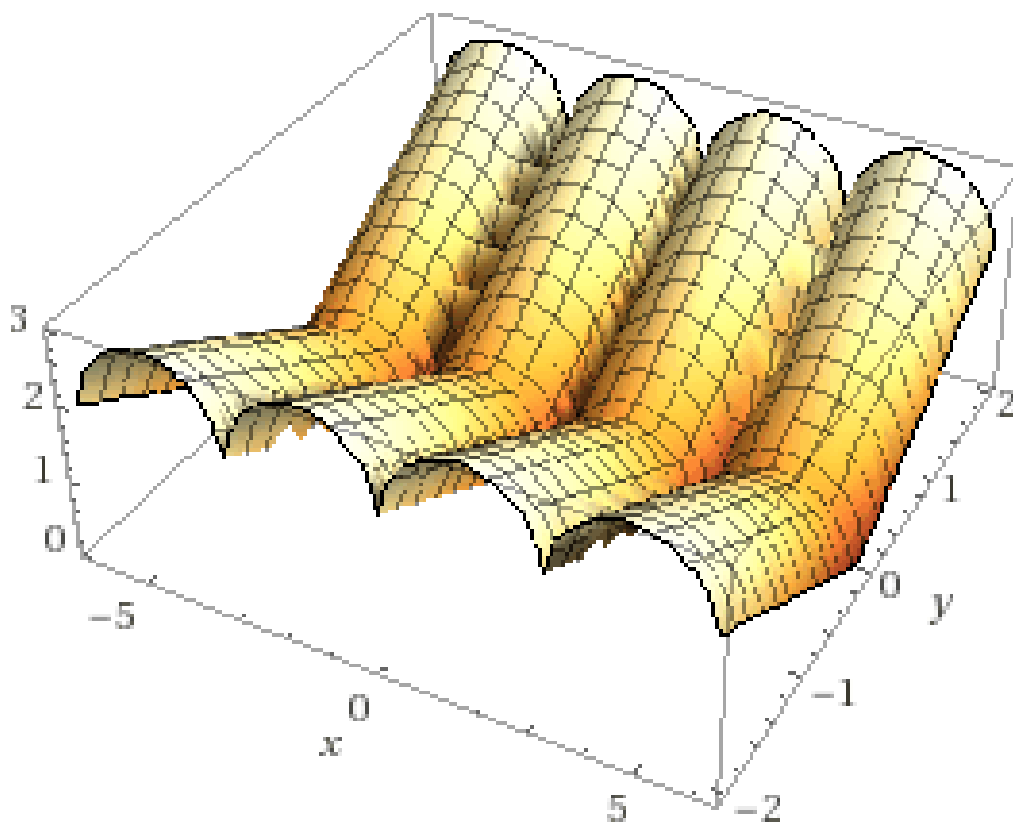


Рисунок 7 График исследуемой на минимум функции

$$4. f(x, y) = \sqrt{|x|} + |\sin y|$$

на брус $[-1; 1] \times [-\frac{\pi}{2}; \frac{\pi}{2}]$, известна её ε -константа Липшица

$$L(\varepsilon) = \frac{1}{4\varepsilon} + 1$$

Таблица 6. Результаты работы алгоритма

Брус	ε^*	ε	$L(\varepsilon)$	h	x_{min}	y_{min}	F_n	n	m	$t, \text{мс}$
$[-1; 1] \times [-\frac{\pi}{2}; \frac{\pi}{2}]$	0.05	0.01	26	0.001538	0.000000	-0.000027	0.000027	1300	2043	122
		0.001	251	0.000195	-0.000088	-0.000063	0.009425	10245	16093	7390
	0.01	0.005	51	0.000098	0.000000	-0.000012	0.000012	20400	32045	29085
		0.001	251	0.000036	0.000004	0.000008	0.002004	55778	87616	217442

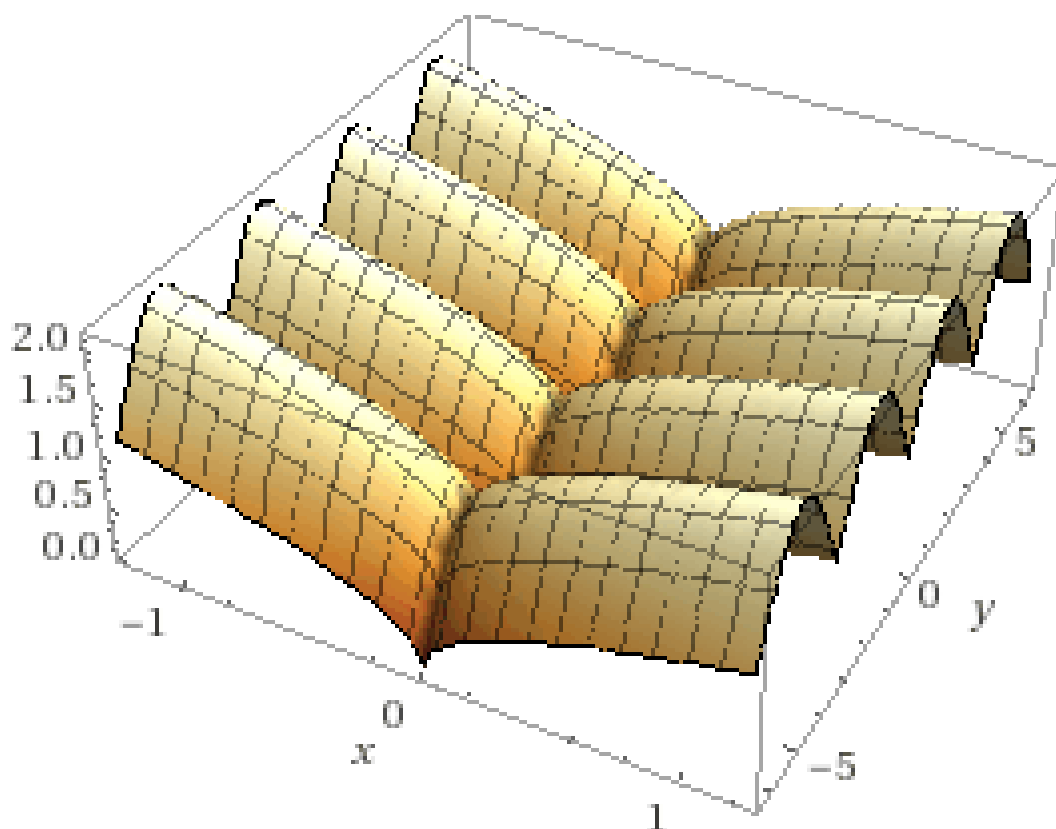


Рисунок 8 График исследуемой на минимум функции

ГЛАВА 4. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМОВ

Обоснование выбора языка и среды программирования

Программная реализация алгоритмов была выполнена на языке программирования C#, поскольку это отлично зарекомендовавший себя на рынке во всём мире язык со множеством удобного функционала, работающего «из коробки». К числу его положительных черт можно отнести:

- Это объектно-ориентированный, мультипарадигмальный, рефлексивный язык программирования.
- C# является «родным» языком программирования для среды Microsoft .Net, потому что он самым тесным и эффективным образом интегрирован с ней.
- При компилировании программы CIL(Common Intermediate Language - «высокоуровневый ассемблер» виртуальной машины .NET) – инструкции распространяются в специальных единицах – сборках, и сопровождаются своими метаданными, что делает её самодостаточным объектом. Благодаря тому, что в исполняемом файле, помимо самой программы, находятся метаданные, можно получать объекты, которые описывают типы, модули и сборки во время исполнения кода (run-time).
- Автоматическое управление памятью.
- Библиотека .NET Framework содержит богатый набор классов для работы с различными видами данными, такими как: ввод / вывод информации, работа с базами данных, работа с сетью, работа с графикой и так далее.
- Для C# кода предоставляется огромный набор «синтаксического сахара», позволяющего создавать лаконичный, понятный для чтения код. Также, сокращая код, необходимый для реализации одних и тех же действий, уменьшается шанс допустить ошибку.

Примерами «синтаксического сахара» являются: лямбда выражения, анонимные типы, LINQ (набор появившихся в .NET функций, которые значительно расширяют возможности синтаксиса языков C# и Visual Basic).

В качестве среды программирования была выбрана MS Visual Studio 2017 Community, как последняя версия Visual Studio на момент начала разработки. Community версия, к тому же, распространяется абсолютно бесплатно и при этом сохраняет весь свой основной функционал от старших версий.

Проектирование программного обеспечения

Разработка любых программных решений начинается с их проектирования. Данный процесс очень важен, так как любая ошибка может привести к тому, что на некотором этапе разработки придётся с нуля пересматривать архитектуру некоторых модулей или всего программного комплекса.

На этапе проектирования разрабатываются архитектура будущего программного обеспечения, устройство его компонентов и пользовательские интерфейсы. В зависимости от рода и сложности создаваемой программы, проектирование может обеспечиваться как «ручным», так и автоматизированным проектированием. Для выражения характеристик программного обеспечения разрабатываются документы и схемы в различных нотациях, такие как ER-диаграммы, UML-диаграммы, блок-схемы, структурные схемы, макеты и другие.

С точки зрения целей будущего использования, программное решение удобнее всего реализовать в формате библиотеки классов (DLL(англ. Dynamic Link Library — «библиотека динамической компоновки») в среде .Net). Далее будут описаны и обоснованы реализации разработанных алгоритмов минимизации.

Реализация алгоритма равномерного перебора отыскания глобального минимума непрерывной (ε -липшицевой) на отрезке функции одной переменной

Для начала, удобнее всего будет представить данное в главе 1 словесное описание алгоритма в виде блок-схемы. Данный вид описания алгоритмов подчиняется Государственному стандарту - ГОСТ 19.701-90 «Схемы алгоритмов программ, данных и систем» [7]. Этот ГОСТ практически полностью соответствует международному стандарту ISO 5807:1985

На рисунке 9 изображена блок-схема алгоритма равномерного перебора отыскания глобального минимума непрерывной на отрезке функции одной переменной.

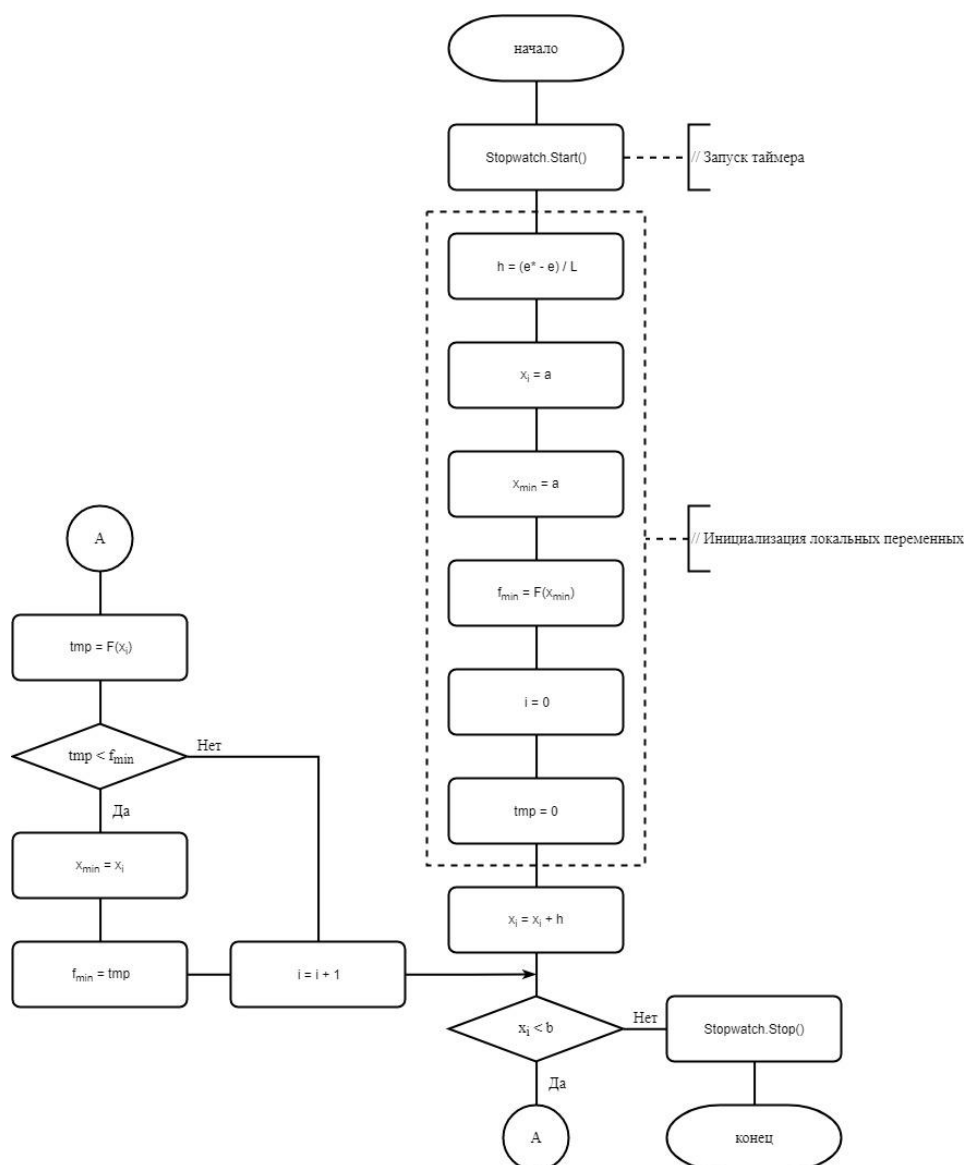


Рисунок 9 Блок-схема алгоритма случая функции одной переменной

По аналогии со стандартным пространством имён Math в среде .Net, было принято решение реализовывать алгоритм в виде статической функции, так как для работы алгоритма нет необходимости хранить никакие данные от запуска к запуску, а входными параметрами и выходными данными являются стандартные структуры среды. Получившийся исходный код функции приведён ниже.

```

    /// <summary>
    /// Модификация метода равномерного перебора поиска глобального
    минимума для эпсилон-липшецевых функций
    /// </summary>
    /// <param name="F">Исследуемая на глобальный минимум
    функция</param>
    /// <param name="a">Левая граница отрезка</param>
    /// <param name="b">Правая граница отрезка</param>
    /// <param name="L">Константа Липшица</param>
    /// <param name="e">Параметр, выбираемый из условия е-
    Липшицевости</param>
    /// <param name="e2">Погрешность, с которой отыскивается приближённое
    значение минимума функции</param>
    /// <returns>
    /// x-значение на оси Ох, в котором достигается глобальный минимум;
    /// F-глобальный минимум переданной функции на рассматриваемом
    отрезке;
    /// n-количество пробных точек(итераций);
    /// time-время, затраченное на выполнение поиска.</returns>
    public static (double h, double x, double F, double n, long time)
    UniformSearchByBiryukov(Func<double, double> F, double a, double b, double L,
    double e, double e2)
    {
        // Таймер для приблизительного измерения производительности
    алгоритма
        var sw = new Stopwatch();
        sw.Start();

        double h = (e2 - e) / L // шаг
            , xi = a // координата на оси Ох, значение функции в которой
    рассматривается на текущей итерации
            , xMin = xi // координата оси Ох, на которой достигается
    лучшее приближение к глобальному минимуму функции на текущей итерации
            , fMin = F(xMin) // лучшее приближение к глобальному минимуму
    функции на текущей итерации

```

```

        , tmp;           // временное хранилище для подмены лучшего
приближения к глобальному минимуму
        int i = 0;

        while ((xi += h) < b)
        {
            // если значение функции в текущей точке меньше последнего
сохранённого - заменяем его
            if ((tmp = F(xi)) < fMin)
            {
                xMin = xi;
                fMin = tmp;
            }
            i++;
        }

        sw.Stop();

        return (h, xMin, fMin, i, sw.ElapsedMilliseconds);
    }

```

Входными параметрами алгоритма являются: исследуемая на минимум функция F , координаты начала a и конца b отрезка, постоянная Липшица L_ε и параметры ε и ε^* .

Результатом выполнения функции являются: шаг h , значение x на оси Ox , в котором достигается глобальный минимум переданной функции, глобальный минимум F функции, количество пробных точек n и затраченное на выполнение алгоритма время $time$.

Реализация алгоритма для случая функции двух переменных

Алгоритм равномерного перебора отыскания глобального минимума непрерывной (ε -липшицевой) на бруске функции двух переменных, что очевидно, очень похож на алгоритм для случая одной переменной.

При его реализации возникает задача выбора построения и обхода сетки, налагаемой на брус. В работе принят следующий алгоритм обхода:

- зафиксировать значение по оси Oy ;

- обработать узлы сети $\forall x \in [a; b]$ при фиксированном значении y ;
- перейти к следующему значению y в соответствии с выбранным шагом.

Это не единственно возможный способ, однако от выбора способа обхода сетки суть алгоритма останется неизменной. На рисунке 10 приведена блок-схема алгоритма для случая функции двух переменных.

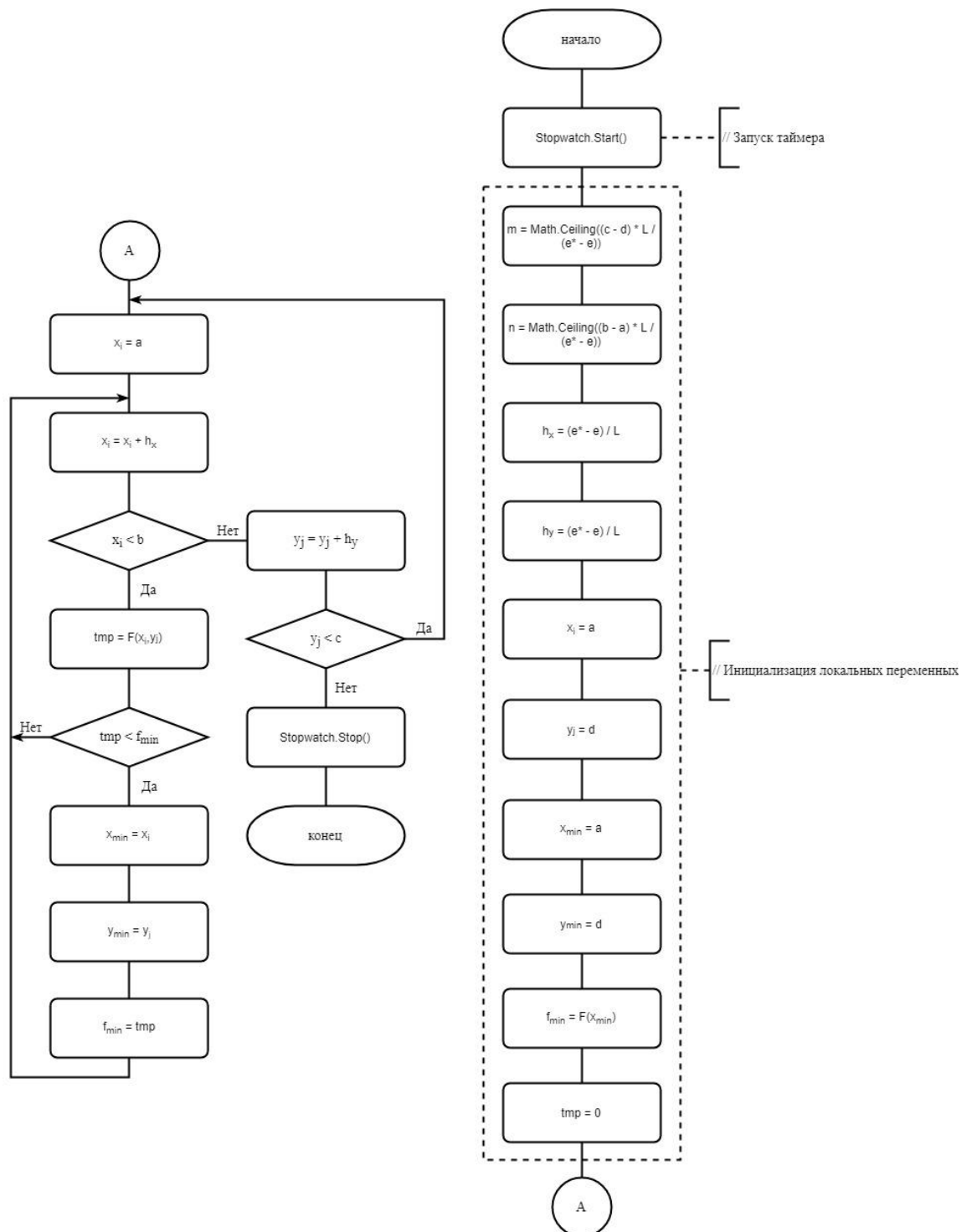


Рисунок 10 Блок-схема алгоритма случая функции двух переменных

Как и для предыдущего случая, алгоритм реализован в виде статической функции по тем же причинам. Причём случай функции двух переменных выполнен как перегрузка метода, решающего случай функции одной переменной, основываясь на принципе полиморфизма. Исходный код приведён ниже.

```

    /// <summary>
    /// Модификация метода равномерного перебора поиска глобального
    минимума для эпсилон-липшецевых функций двух переменных
    /// </summary>
    /// <param name="F">Исследуемая на глобальный минимум
    функция</param>
    /// <param name="a">Левая граница бруса</param>
    /// <param name="b">Правая граница бруса</param>
    /// <param name="d">Нижняя граница бруса</param>
    /// <param name="c">Верхняя граница бруса</param>
    /// <param name="L">Константа Липшица</param>
    /// <param name="e">Параметр, выбираемый из условия е-
    Липшицевости</param>
    /// <param name="e2">Погрешность, с которой отыскивается приближённое
    значение минимума функции</param>
    /// <returns>
    /// х-значение на оси Ох, в котором достигается глобальный минимум;
    /// у-значение на оси Оу, в котором достигается глобальный минимум;
    /// F-глобальный минимум переданной функции на рассматриваемом
    отрезке;
    /// n-количество пробных точек по горизонтали;
    /// n-количество пробных точек по вертикали;
    /// time-время, затраченное на выполнение поиска.</returns>
    public static (double hx, double hy, double x, double y, double F, double n,
    double m, long time) UniformSearchByBiryukov(Func<double, double, double> F,
    double a, double b, double d, double c, double L, double e, double e2)
    {
        // Таймер для приблизительного измерения производительности
    алгоритма
        var sw = new Stopwatch();
        sw.Start();

        double m = Math.Ceiling((c - d) * L / (e2 - e)), // число минимально
    необходимых пробных точек по оси Оу
        n = Math.Ceiling((b - a) * L / (e2 - e)); // число минимально
    необходимых пробных точек по оси Ох

```

```

double hx = (e2 - e) / L // шаг по оси Oх
    , hy = (e2 - e) / L // шаг по оси Oy
    , xi = a // координата на оси Oх, значение функции в которой
рассматривается на текущей итерации
    , yi = d // координата на оси Oy, значение функции в которой
рассматривается на текущей итерации
    , xMin = xi // координата оси Oх, на которой достигается
лучшее приближение к глобальному минимуму функции на текущей итерации
    , yMin = yi // координата оси Oy, на которой достигается
лучшее приближение к глобальному минимуму функции на текущей итерации
    , fMin = F(xMin, yMin) // лучшее приближение к глобальному
минимуму функции на текущей итерации
    , tmp; // временное хранилище для подмены лучшего
приближения к глобальному минимуму

```

```

do
{
    xi = a;
    while ((xi += hx) < b)
    {
        // если значение функции в текущей точке меньше последнего
сохранённого - заменяем его
        if ((tmp = F(xi, yi)) < fMin)
        {
            xMin = xi;
            yMin = yi;
            fMin = tmp;
        }
    }
}
while ((yi += hy) < c);
sw.Stop();
return (hx, hy, xMin, yMin, fMin, n, m, sw.ElapsedMilliseconds);
}

```

Входными параметрами алгоритма являются: границы бруса $[a; b] \times [d; c]$, постоянная Липшица L_ε и параметры ε и ε^* .

Результатом выполнения функции являются: шаг h_x по оси Ox , шаг h_y по оси Oy , значения x на оси Ox и y на оси Oy , в которых достигается глобальный минимум переданной функции, глобальный минимум F функции, количество пробных точек n по горизонтали и m по вертикали, и затраченное на выполнение алгоритма время $time$.

ГЛАВА 5. ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

Основные понятия и принципы тестирования программного продукта

Тестирование любого программного продукта, вне зависимости от его назначения или объёма исходного кода, является важной частью разработки программного продукта. Без полноценно проведённого тестирования невозможно оценить, насколько качественно программный продукт выполнен. Чем качественнее будут проведены тесты, тем меньше вероятность, что будут упущены ошибки, которые в будущем придётся исправлять

Тестирование — это процесс испытания, исследования программного продукта, имеющий следующие основные цели:

- показать, что программный продукт выполнен качественно и в соответствии со всеми требованиями;
- обнаружить такие ситуации, при возникновении которых поведение программы является некорректным, нежелательным или вовсе непредсказуемым.

Информационные потоки процесса тестирования изображены на рисунке 11.

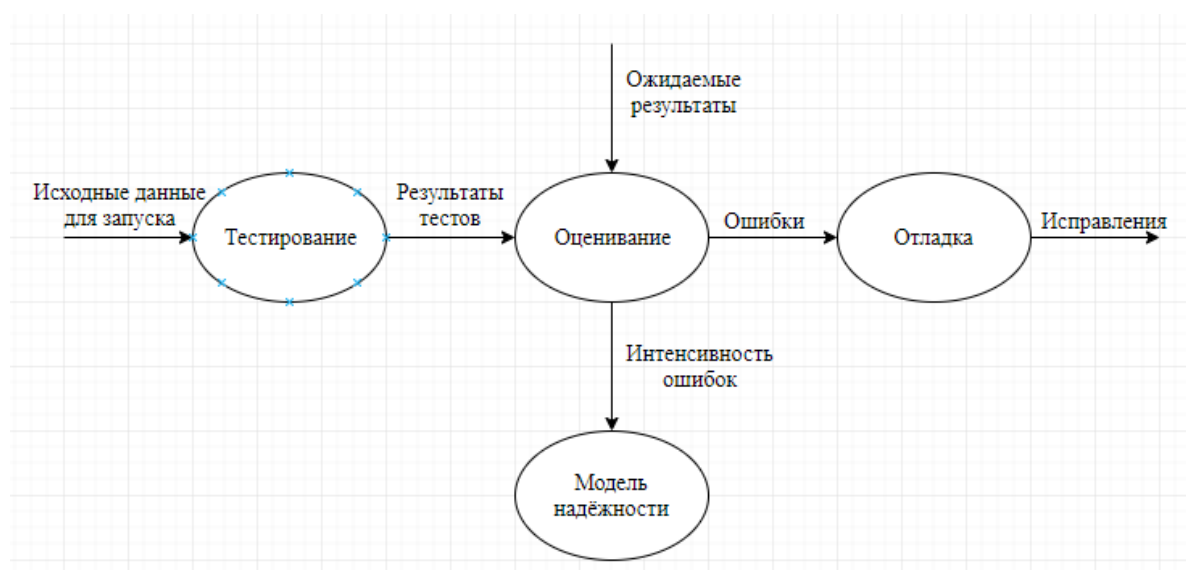


Рисунок 11 Информационные потоки процесса тестирования

В тех случаях, когда программный продукт разрабатывается для массового применения, отсутствие должного тестирования, а в следствии, упущение ряда ошибок, может вызвать негативное отношение к нему среди пользователей.

Сколь хорошо не были бы проведены тесты, их результаты не могут однозначно указывать на полное отсутствие возможности возникновения ошибок. Для наиболее полного определения качества программного обеспечения производится анализ совокупности следующих составляющих:

- надёжность;
- сопровождаемость;
- практичность;
- эффективность;
- мобильность;
- функциональность.

В процессе тестирования для каждого модуля программного обеспечения создаются наборы тестов с вариациями исходных значений и ожидаемыми результатами отработки модулей.

По выполнению каждого теста результаты их исполнения проверяются или, другими словами, реальные результаты исполнения тестов сравниваются с ожидаемыми, заранее подготовленными и проверенными результатами. При обнаружении несовпадений, фиксируется ошибка. Каждая найденная ошибка обязательно должна быть исправлена, поэтому начинается процесс поиска причин неверной работы модулей программного продукта и его отладка. На поиск места дефекта и исправление может потребоваться от пары минут до нескольких дней. Эта неопределенность приводит к затруднению планирования действий.

По окончании отладки на исправленные места исходных кодов пишутся новые, либо исправляются старые тесты, и процесс проверки начинается заново.

Если функции программного обеспечения реализованы правильно, а обнаруженные ошибки легко исправляются, может быть сделан один из двух выводов:

- качество и надежность ПО удовлетворительны;
- тесты не способны обнаруживать серьезные ошибки.

Существуют 2 принципа тестирования программы:

- функциональное тестирование (тестирование «черного ящика»);
- структурное тестирование (тестирование «белого ящика»). [8]

Тестирование, применённое к разрабатываемому программному обеспечению

На данный момент разработано большое число различных методологий проведения тестирования программных продуктов, такие как функциональное, нагрузочное, интеграционное и другие тестирования. Однако все они подразумевают если не понимание «изнанки» процессов, то, как минимум, наличие этих самых процессов, на которые по окончании или во время разработки пишутся тесты. Однако существует другая методология, основанная на том, что первым делом разрабатываются именно тестовые наборы, а лишь потом пишется код, ими проверяемый. Такой подход, как никакой другой подходит для процесса разработки математической библиотеки, когда заранее известны численные примеры с их точными решениями. Такой подход используется в методологии TDD.

Разработка через тестирование (TDD, test-driven development) - это процесс разработки программного обеспечения, который основывается на повторении очень короткого цикла разработки: сначала разработчик пишет (изначально неудачный) автоматический тестовый пример, который определяет желаемое улучшение или новую функцию, затем выдает минимальное количество кода, чтобы пройти этот тест, и, наконец, рефакторит новый код в соответствии с приемлемыми стандартами. [9]

Обычно выполняется следующая последовательность шагов:

- Добавить тест;
- Запустить все тесты и посмотреть, не сработает ли новый;
- Написать код;
- Запустить тесты;
- Рефакторинг кода;
- Повторение.

Реализация тестовых примеров

Исходя из сказанного в предыдущем пункте, первым шагом при разработке программного продукта было создание автоматизированных тестов на основании рассчитанных заранее численных примеров. Однако эти тесты необходимо ещё реализовать. Тут на помощь может прийти любая из существующих библиотек разработки автоматизированных тестов: NUnit, xUnit, JUnit или даже встроенная в Visual Studio система построения и исполнения тестов.

В рамках данной дипломной работы для реализации тестовых примеров был использован фреймворк xUnit совместно с библиотекой FluentAssertions. Оба этих средства доступны для скачивания и внедрения в своё решение через NuGet - систему управления пакетами для платформ разработки Microsoft.

xUnit.net - это бесплатный инструмент для модульного тестирования в среде .NET Framework с открытым исходным кодом. Написанная первоначальным изобретателем NUnit v2, xUnit - это новейшая технология для модульного тестирования C #, F #, VB.NET и других языков .NET. xUnit работает с ReSharper, CodeRush, TestDriven.NET и Xamarin. Он является частью .NET Foundation и действует в соответствии с их кодексом поведения. Лицензируется под Apache 2 (лицензия, утвержденная OSI) [10].

FluentAssertions – библиотека, предоставляющая обширный набор методов расширений, которые позволяют более естественно указать ожидаемый результат модульных тестов в стиле TDD и BDD.

Связка двух вышеуказанных инструментов позволяет очень удобно и продуктивно работать над написанием автоматизированных модульных тестов.

Пример одного из тестов, который на самом деле при запуске исполнительной среды xUnit разворачивается в три различных теста, приведён ниже.

```
[Theory]
[InlineData(-Math.PI, 3 * Math.PI, 0.001, 0.01)]
[InlineData(-Math.PI / 2.0, Math.PI / 2.0, 0.0005, 0.001)]
[InlineData(0, Math.PI, 0.01, 0.1)]
public void OneDimensionF1(double a, double b, double e, double e2)
{
    // Рассматриваемая функция
    double F(double x)
        => Math.Abs(x) + Math.Sqrt(Math.Abs(Math.Sin(x)));

    // ожидаемый результат
    double fMin = 0;

    // полученное решение
    var result = MathStrategy.UniformSearchByBiryukov(
        F,
        a, b, // [a;b]
        L(e), // L=L(e)=1/(4e)
        e, e2); // e, e*

    // Проверка прохождения теста
    (result.F - fMin).Should().BeLessOrEqualTo(e2);
}
```

Атрибут [Theory] обозначает параметризованный тест, который является истинным для подмножества данных. Эти данные могут быть предоставлены несколькими способами, но наиболее распространённым является атрибут [InlineData]. Таким образом происходит выполнение тела одного и того же модульного теста на различных наборах исходных данных.

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Руководство по эксплуатации

Программный продукт является библиотекой DLL и предоставляет возможности нахождения глобального минимума непрерывной (ε -липшицевой) функции одной переменной на отрезке или двух переменных на бруске приведёнными и обоснованными ранее в работе методами.

Чтобы воспользоваться разработанной библиотекой, необходимо открыть обозреватель решения, в котором она будет использоваться. Далее – развернуться проект и нажать правой кнопкой на Ссылки – Добавить ссылку (смотри рисунок 12).

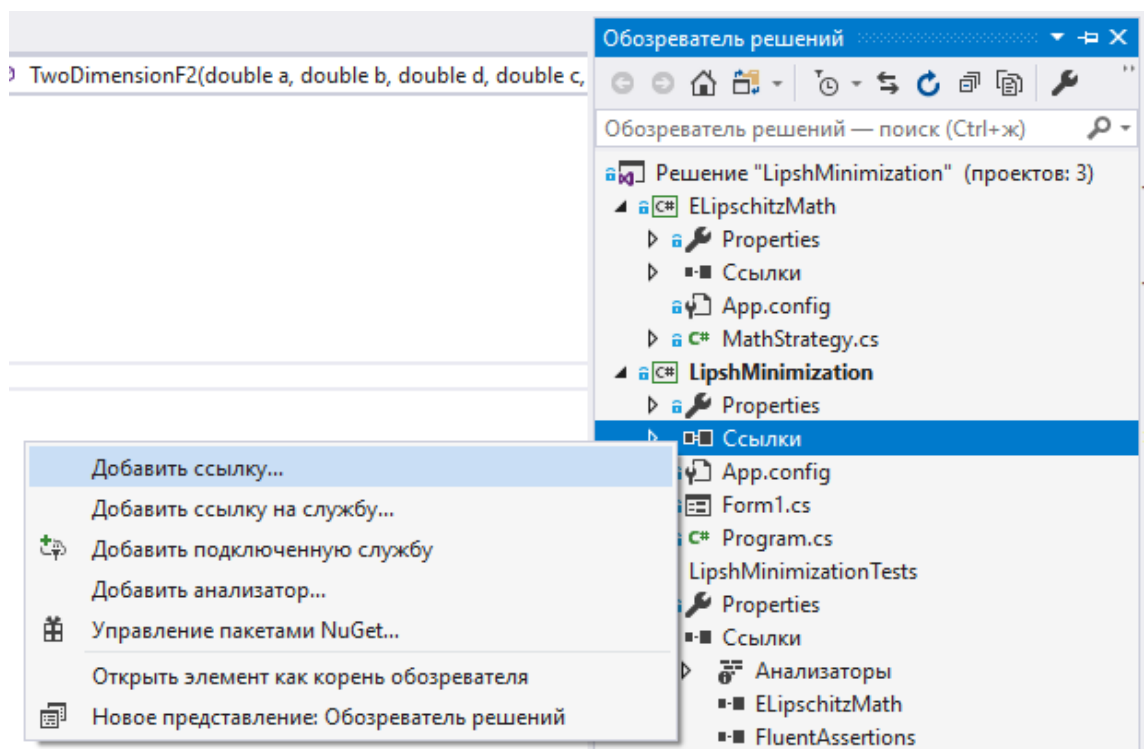


Рисунок 12 Добавление ссылки к проекту

В открывшемся дочернем окне необходимо справа выбрать подпункт Обзор и нажать в правом нижнем углу кнопку «Обзор...», после чего указать путь до файла математической библиотеки LipshMinimization.ELipschitzMath.dll. Подтверждение выбора файла, на который будет установлена ссылка, производится нажатием кнопки Добавить (рисунок 13).

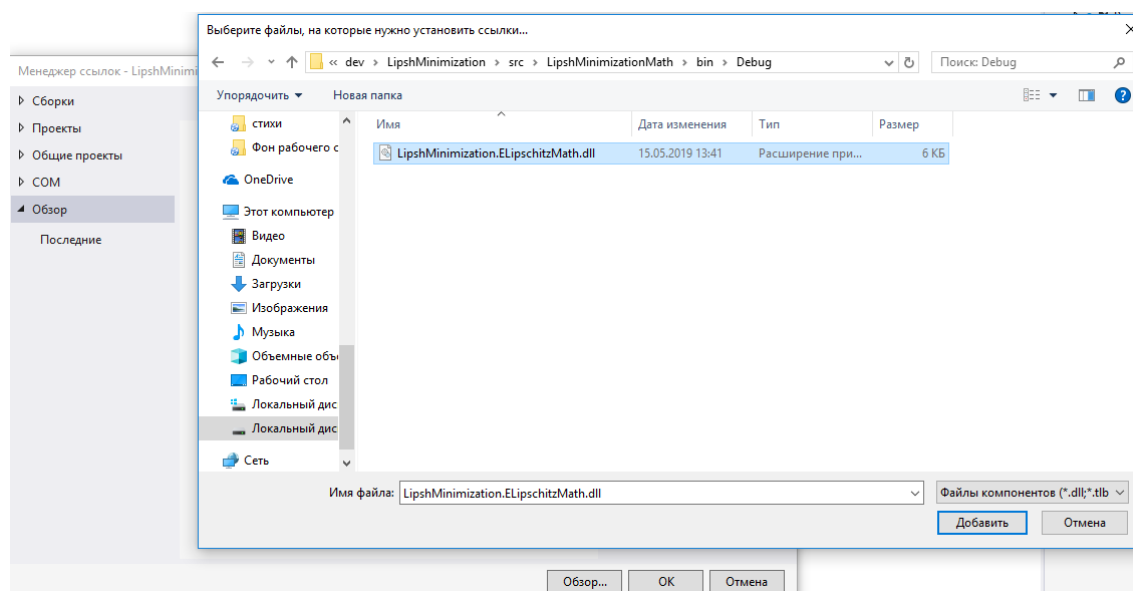


Рисунок 13 Выбор файла, на который добавляется ссылка

После добавления ссылки на .dll файл, в коде достаточно указать using на пространство имён разработанной библиотеки (рисунок 14) и можно будет использовать предоставляемый функционал.

```
using LipshMinimization.ELipschitzMath;
```

Рисунок 14 Подключение пространства имён

Вызов непосредственно функций, выполняющих минимизацию задаваемой функции, производится обращением к соответствующей перегрузке метода UniformSearchByBiryukov (смотри рисунок 15).

```
var result = MathStrategy.UniformSearchByBiryukov(F, xmin, xmax, L, e, e2);
```

Ⓢ (double h, double x, double F, double n, long time) MathStrategy.UniformSearchByBiryukov(Func<double, double> F, double a, double b, double L, double e, double e2) (+ 1 перегрузка)
 Модификация метода равномерного перебора поиска глобального минимума для эпсилон-липшецевых функций

Рисунок 15 Вызов перегрузки метода, выполняющего минимизацию функции одной переменной

Вспомогательная программа (Визуализатор)

При написании работы был использован разработанный автором вспомогательный программный продукт, выполняющий следующие функции:

- визуализация графика исследуемой функции (только для случая функции одной переменной);
- решение задачи минимизации указанной функции с помощью вызова методов, разработанных в основном программном продукте – математической библиотеке.

Данная программа не является полноценным пользовательским приложением, так как для её использования необходимо внедряться в исходный код – изменять функцию и входные параметры алгоритма минимизации.

```
public partial class Form1 : Form
{
    private const float xmin = (float)(-2 * Math.PI);
    private const float xmax = (float)(3 * Math.PI);
    private const float ymin = -1;
    private const float ymax = 5;

    /// <summary>
    /// Функция, по которой строится график
    /// </summary>
    private double F(double x)
        => Math.Sqrt(Math.Abs(x)) + Math.Abs(Math.Sin(x));

    /// <summary>
    /// Конструктор по умолчанию
    /// </summary>
    public Form1()
    {
        InitializeComponent();

        this.FormBorderStyle = FormBorderStyle.FixedSingle;

        this.Axis();
        this.Plot(F, Color.Red);

        double e = 0.001
            , e2 = 0.01
            , L = 1.0 / (4.0 * e) + 1;

        var result = MathStrategy.UniformSearchByBiryukov(F, xmin, xmax, L, e, e2);
    }
}
```

Рисунок 16 Часть исходного кода, подлежащего изменению при повторном использовании программы визуализации

После внесения изменений, связанных с решаемым численным примером, необходимо выполнить пересборку проекта. Это приведёт к созданию нового исполняемого файла LipshMinimization.exe, включающего внесённые правки. Запустить программу можно двойным щелчком по исполняемому файлу, либо нажатием клавиши F5 в среде Visual Studio при открытом проекте.

Программа выполнит решение поставленной задачи минимизации и выведет диалоговое окно с результатами (смотри рисунок 17).

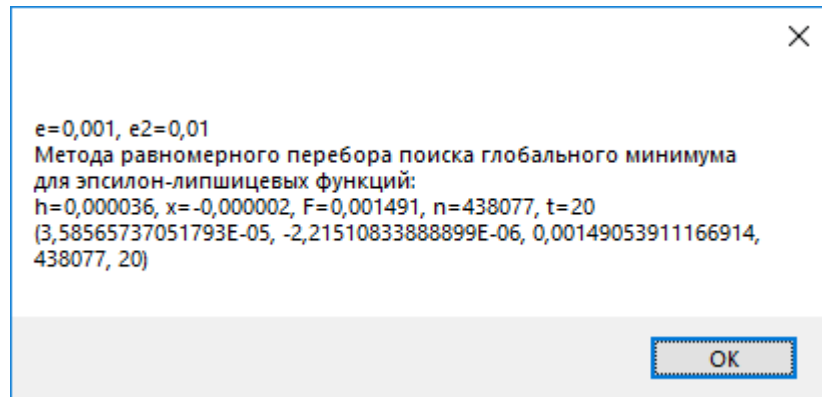


Рисунок 17 Результаты вычисления глобального минимума с применением разработанной математической библиотеки

В этом диалоговом окне отображаются:

- Значения параметров ε и ε^* ;
- Наименование метода;
- Результаты работы алгоритма с округлением до 6 знака после запятой;
- Точные результаты работы алгоритма.

Для того чтобы увидеть график функции, для которой производилась минимизация, необходимо нажать кнопку ОК. Откроется главное окно программы, в котором располагается график функции в пределах исследуемой области (рисунок 18).

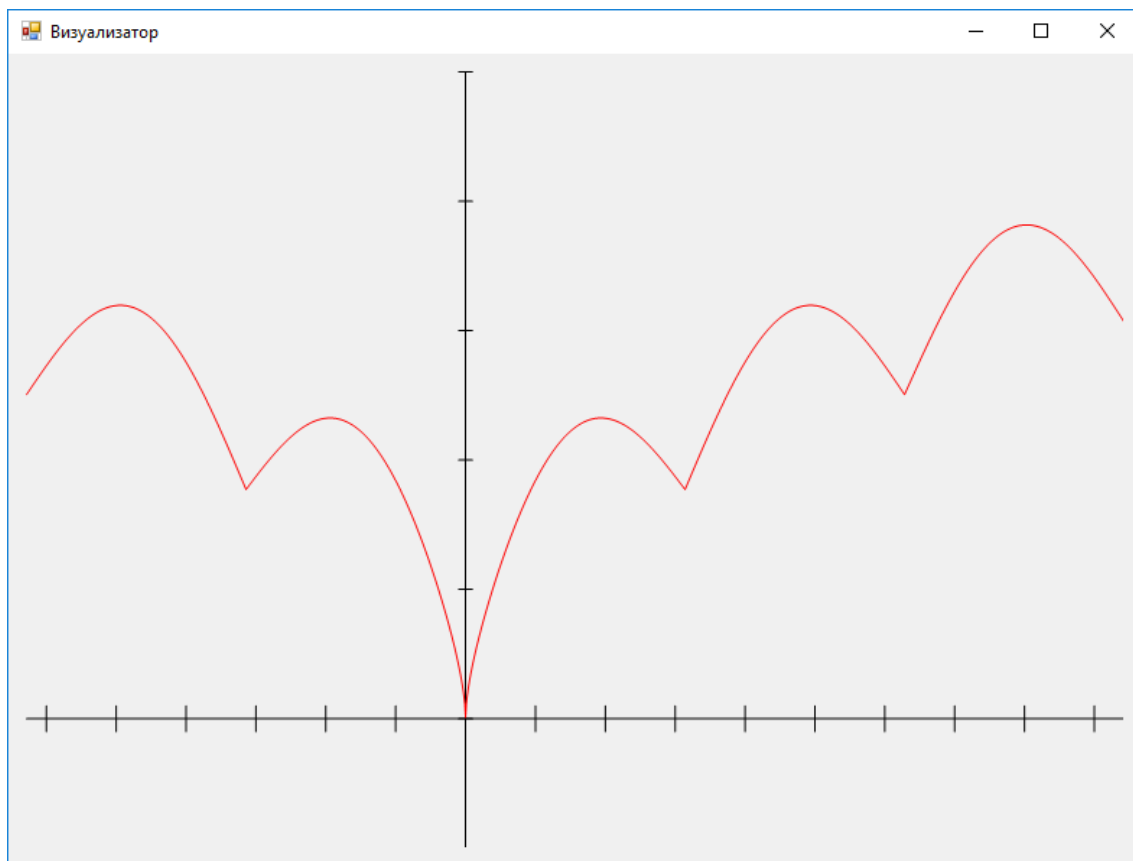


Рисунок 18 Основное окно программы Визуализатор с графиком исследуемой функции

Для завершения работы Визуализатора достаточно нажать крестик в правом верхнем углу главного окна.

ЗАКЛЮЧЕНИЕ

В рамках проделанной работы

- изучено современное состояние методов приближённого вычисления минимального значения ε -липшицевых на выпуклом компакте функций;
- разработаны, описаны и обоснованы следующие численные методы:
 - алгоритм приближённого поиска глобального минимума непрерывной на отрезке функции, основанный на идеях равномерного перебора;
 - алгоритм приближённого поиска глобального минимума непрерывной на бруске функции двух переменных, также основанный на идее равномерного перебора по сетке.

Следует подчеркнуть, что в обоих случаях на функции не накладывались никакие дополнительные требования, кроме их непрерывности на выпуклом компакте.

- разработано программное обеспечение для обоих алгоритмов;
- с помощью разработанного программного обеспечения рассчитана серия тестовых задач. Для некоторых ранее известных результатов поиска глобального минимума проведено сравнение с полученными предложенными методами, которое показало хорошее совпадение приближённых минимальных значений функции.

Разработанные методы могут быть полезны при решении задач негладкого математического программирования, когда минимизируемая функция является достаточно «плохой» - не является выпуклой или липшицевой.

Таким образом, полностью решены все задачи, поставленные научным руководителем.

CONCLUSION

As part of the work done

- The state of the art of approximate calculation of the minimum value of ε -Lipschitz functions on convex compact functions has been studied;
- The following numerical methods have been developed, described and proved:
 - an algorithm for the approximate search for a global minimum of a continuous function on an interval, based on the ideas of uniform search;
 - an algorithm for the approximate search for a global minimum of a continuous function of two variables on a rectangle, also based on the idea of a uniform search over the grid.

It should be noted that in both cases no additional requirements were imposed on the functions, except for their continuity on the convex compact.

- developed software for both algorithms;
- with the help of the developed software, a series of test problems was calculated. For some previously known results of the search for a global minimum, a comparison was made with the obtained proposed methods, which showed good agreement between the approximate minimum values of the function.

The developed methods can be useful in solving problems of non-smooth mathematical programming when the function being minimized is rather “bad” - it is not convex or Lipschitz.

Thus, all the tasks posed by the academic advisor are completely solved.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Vanderbei, R.J. Extension of Piyavskii's Algorithm to Continuous Global Optimization / R.J. Vanderbei // Journal of Global Optimization. – 1999. – Vol. 14. – P. 205-216.
2. Заботин, В.И. Алгоритм вычисления минимальной оценки ε -постоянной Липшица непрерывной функции / В.И. Заботин, П.А. Чернышевский // Вестник КГТУ им. А.Н. Туполева. – 2018. - № 2, вып. 2. – С. 127-132.
3. Заботин, В.И. Два алгоритма отыскания проекции точки на невыпуклое множество в нормированном пространстве / В.И. Заботин, Н.К. Арутюнова // Журнал вычислительной математики и математической физики. – 2013. – Т. 53, № 3. – С. 344-349.
4. Арутюнова, Н.К. Алгоритмы проектирования точки на поверхность уровня непрерывной на компакте функции / Н.К. Арутюнова, А.М. Дуллиев, В.И. Заботин // Журнал вычислительной математики и математической физики. – 2014. – Т. 54, № 9. – С. 1448-1454.
5. Арутюнова, Н.К. Модификация метода Евтушенко поиска глобального минимума для случая непрерывной на компакте функции / Н.К. Арутюнова // Вестник КГТУ им. А.Н. Туполева. – 2013. - № 2, вып. 2. – С. 154-157.
6. N.K. Arutyunova Models and methods for three external ballistics inverse problems / N.K. Arutyunova, A.M. Dulliev, V.I. Zabotin // , Vestnik YuUrGU. – 2017. – Vol. 10. – P. 78-91.
7. ГОСТ 19.701–90. Единая система программной документации. – Москва: Стандартинформ, 2010. - 22 с.

8. Технологии разработки программного обеспечения. Глава 6. Структурное тестирование программного обеспечения [Электронный ресурс] – Режим доступа : <http://textarchive.ru/c-1144105-p14.html>.
9. Technology Conversations. Test Driven Development (TDD): Example Walkthrough. [Электронный ресурс] – Режим доступа : <https://technologyconversations.com/2013/12/20/test-driven-development-tdd-example-walkthrough/>
- 10.xUnit.net. About xUnit.net. [Электронный ресурс] – Режим доступа : <https://xunit.net/>

ПРИЛОЖЕНИЕ 1. Письмо с подтверждением принятия публикации



Всероссийская молодежная научно-практическая конференция

УНИИД ПГУ им. Шолом-Алейхема <uniid@bk.ru>

Кому: aleks-herou@mail.ru

6 мая, 2:13 1 файл

Добрый день!

Ваша статья прошла рецензирование и принята к публикации в сборнике трудов конференции.

Так же направляю электронный вариант сертификата участника.

С уважением,

инженер отдела научно-исследовательской деятельностью

управления научно-исследовательской

и инновационной деятельностью

ФГБОУ ВО "ПГУ им. Шолом-Алейхема"

Дернова Ксения,

контактный телефон: [+7 914 427-78-61](tel:+79144277861), [+7\(42622\) 4-63-69](tel:+74262246369)

uniid@bk.ru

ПРИЛОЖЕНИЕ 2. Сертификат участника конференции



ПРИЛОЖЕНИЕ 3. Листинг программного продукта

```
using System;
using System.Diagnostics;

namespace LipshMinimization.ELipschitzMath
{
    public static class MathStrategy
    {
        /// <summary>
        /// Модификация метода Евтушенко поиска глобального минимума для случая непрерывной
на отрезке функции
        /// </summary>
        /// <param name="F">Исследуемая на глобальный минимум функция</param>
        /// <param name="a">Левая граница отрезка</param>
        /// <param name="b">Правая граница отрезка</param>
        /// <param name="L">Константа Липшица</param>
        /// <param name="e">Параметр, выбираемый из условия е-Липшицевости</param>
        /// <param name="e2">Погрешность, с которой отыскивается приближённое значение
минимума функции</param>
        /// <returns>
        /// x-значение на оси Oх, в котором достигается глобальный минимум;
        /// F-глобальный минимум переданной функции на рассматриваемом отрезке;
        /// n-количество пробных точек(итераций);
        /// time-время, затраченное на выполнение поиска.</returns>
        public static (double x, double F, double n, long time)
EvtushenkoMethodByArytunova(Func<double, double> F, double a, double b, double L, double e,
double e2) // модифицированный Арутюновой метод Евтушенко
        {
            // Таймер для приблизительного измерения производительности алгоритма
            var sw = new Stopwatch();
            sw.Start();

            double h = 2.0 * (e2 - e) / L
                , xi
                , Fmin
                , xMin;

            // получение xi+1
            double NextX(double x)
                => x + h + (F(x) - Fmin) / L;

            double exitParam = b - h / 2.0;

            double xi_1 = xMin = a + h / 2.0, tmp = 0;
            Fmin = F(xi_1);
            int i = 1;
            do
            {
                xi = xi_1;

                tmp = Math.Min(Fmin, F(xi));
                if(tmp!=Fmin)
                {
                    Fmin = tmp;
                    xMin = xi;
                }

                xi_1 = NextX(xi);
                i++;
            } while (!(xi < exitParam && exitParam <= xi_1));
        }
    }
}
```

```

        xi = Math.Min(xi_1, b);
        tmp = Math.Min(Fmin, F(xi));
        if (tmp != Fmin)
        {
            Fmin = tmp;
            xMin = xi;
        }

        sw.Stop();

        // xi_1 уже хранит xn.
        return (xMin, Math.Min(Fmin, F(xi_1)), i, sw.ElapsedMilliseconds);
    }

    /// <summary>
    /// Модификация метода равномерного перебора поиска глобального минимума для
    эpsilon-липшицевых функций
    /// </summary>
    /// <param name="F">Исследуемая на глобальный минимум функция</param>
    /// <param name="a">Левая граница отрезка</param>
    /// <param name="b">Правая граница отрезка</param>
    /// <param name="L">Константа Липшица</param>
    /// <param name="e">Параметр, выбираемый из условия e-Липшицевости</param>
    /// <param name="e2">Погрешность, с которой отыскивается приближённое значение
    минимума функции</param>
    /// <returns>
    /// x-значение на оси Oх, в котором достигается глобальный минимум;
    /// F-глобальный минимум переданной функции на рассматриваемом отрезке;
    /// n-количество пробных точек (итераций);
    /// time-время, затраченное на выполнение поиска.</returns>
    public static (double h, double x, double F, double n, long time)
    UniformSearchByBiryukov(Func<double, double> F, double a, double b, double L, double e,
    double e2)
    {
        // Таймер для приблизительного измерения производительности алгоритма
        var sw = new Stopwatch();
        sw.Start();

        double h = (e2 - e) / L // шаг
        , xi = a // координата на оси Oх, значение функции в которой
    рассматривается на текущей итерации
        , xMin = xi // координата оси Oх, на которой достигается лучшее
    приближение к глобальному минимуму функции на текущей итерации
        , fMin = F(xMin) // лучшее приближение к глобальному минимуму функции
    на текущей итерации
        , tmp; // временное хранилище для подмены лучшего
    приближения к глобальному минимуму
        int i = 0;

        while ((xi += h) < b)
        {
            // если значение функции в текущей точке меньше последнего сохранённого -
    заменяем его
            if ((tmp = F(xi)) < fMin)
            {
                xMin = xi;
                fMin = tmp;
            }
            i++;
        }

        sw.Stop();
    }

```

```

        return (h, xMin, fMin, i, sw.ElapsedMilliseconds);
    }

    /// <summary>
    /// Модификация метода равномерного перебора поиска глобального минимума для
    эpsilon-липшицевых функций двух переменных
    /// </summary>
    /// <param name="F">Исследуемая на глобальный минимум функция</param>
    /// <param name="a">Левая граница бруса</param>
    /// <param name="b">Правая граница бруса</param>
    /// <param name="d">Нижняя граница бруса</param>
    /// <param name="c">Верхняя граница бруса</param>
    /// <param name="L">Константа Липшица</param>
    /// <param name="e">Параметр, выбираемый из условия e-Липшицевости</param>
    /// <param name="e2">Погрешность, с которой отыскивается приближённое значение
    минимума функции</param>
    /// <returns>
    /// x-значение на оси Oх, в котором достигается глобальный минимум;
    /// y-значение на оси Oу, в котором достигается глобальный минимум;
    /// F-глобальный минимум переданной функции на рассматриваемом отрезке;
    /// n-количество пробных точек по горизонтали;
    /// n-количество пробных точек по вертикали;
    /// time-время, затраченное на выполнение поиска.</returns>
    public static (double hx, double hy, double x, double y, double F, double n, double
m, long time) UniformSearchByBiryukov(Func<double, double, double> F, double a, double b,
double d, double c, double L, double e, double e2)
    {
        // Таймер для приблизительного измерения производительности алгоритма
        var sw = new Stopwatch();
        sw.Start();

        double m = Math.Ceiling((c - d) * L / (e2 - e)), // число минимально
        необходимых пробных точек по оси Oу
        n = Math.Ceiling((b - a) * L / (e2 - e)); // число минимально
        необходимых пробных точек по оси Oх

        double hx = (e2 - e) / L // шаг по оси Oх
        , hy = (e2 - e) / L // шаг по оси Oу
        , xi = a // координата на оси Oх, значение функции в которой
        рассматривается на текущей итерации
        , yi = d // координата на оси Oу, значение функции в которой
        рассматривается на текущей итерации
        , xMin = xi // координата оси Oх, на которой достигается лучшее
        приближение к глобальному минимуму функции на текущей итерации
        , yMin = yi // координата оси Oу, на которой достигается лучшее
        приближение к глобальному минимуму функции на текущей итерации
        , fMin = F(xMin, yMin) // лучшее приближение к глобальному минимуму функции
        на текущей итерации
        , tmp; // временное хранилище для подмены лучшего
        приближения к глобальному минимуму

        do
        {
            xi = a;

            while ((xi += hx) < b)
            {
                // если значение функции в текущей точке меньше последнего сохранённого
                - заменяем его
                if ((tmp = F(xi, yi)) < fMin)
                {
                    xMin = xi;
                    yMin = yi;
                    fMin = tmp;

```

```

        }
    }
}
while ((yi += hy) < c);

sw.Stop();

return (hx, hy, xMin, yMin, fMin, n, m, sw.ElapsedMilliseconds);
}
}
}

```

ПРИЛОЖЕНИЕ 4. Листинг вспомогательного программного продукта (Визуализатор)

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Windows.Forms;

using LipshMinimization.ELipschitzMath;

namespace LipshMinimization
{
    public partial class Form1 : Form
    {
        private const float xmin = (float)(-2 * Math.PI);

        private const float xmax = (float)(3 * Math.PI);

        private const float ymin = -1;

        private const float ymax = 5;

        /// <summary>
        /// Функция, по которой строится график
        /// </summary>
        private double F(double x)
            => Math.Sqrt(Math.Abs(x)) + Math.Abs(Math.Sin(x));

        /// <summary>
        /// Конструктор по умолчанию
        /// </summary>
        public Form1()
        {
            InitializeComponent();

            this.FormBorderStyle = FormBorderStyle.FixedSingle;

            this.Axis();
            this.Plot(F, Color.Red);

            double e = 0.001
                , e2 = 0.01
                , L = 1.0 / (4.0 * e) + 1;

            var result = MathStrategy.UniformSearchByBiryukov(F, xmin, xmax, L, e, e2);
            MessageBox.Show($"e={e}, e2={e2}\nМетода равномерного перебора поиска  
глобального минимума для эпсилон-липшицевых функций:\nh={result.h.ToString("F6")},  
x={result.x.ToString("F6")}, F={result.F.ToString("F6")}, n={result.n},  
t={result.time}\n{result}");
        }

        /// <summary>
        /// Построение осей графика
        /// </summary>
        private void Axis()
        {
            var width = this.picGraph.ClientSize.Width;
            var height = this.picGraph.ClientSize.Height;
            var bitmap = new Bitmap(width, height);

            using (var graphics = Graphics.FromImage(bitmap))
```



```

    {
        graphics.SmoothingMode = SmoothingMode.HighQuality;

        // Transform to map the graph bounds to the Bitmap.
        var rect = new RectangleF(xmin, ymin, xmax - xmin, ymax - ymin);
        PointF[] points =
        {
            new PointF(0, height),
            new PointF(width, height),
            new PointF(0, 0),
        };
        graphics.Transform = new Matrix(rect, points);

        using (var pen = new Pen(Color.Black, 0))
        {
            graphics.DrawLine(pen, xmin, 0, xmax, 0);
            graphics.DrawLine(pen, 0, ymin, 0, ymax);

            for (int x = (int)xmin; x <= xmax; x++)
                graphics.DrawLine(pen, x, -0.1f, x, 0.1f);

            for (int y = (int)ymin; y <= ymax; y++)
                graphics.DrawLine(pen, -0.1f, y, 0.1f, y);
        }
    }

    // Display the result.
    this.picGraph.Image = bitmap;
}

/// <summary>
/// Построение графика указанной функции
/// </summary>
private void Plot(Func<double, double> function, Color? plotColor = null)
{
    // Make the Bitmap.
    var width = this.picGraph.ClientSize.Width;
    var height = this.picGraph.ClientSize.Height;
    var bitmap = new Bitmap(this.picGraph.Image);

    using (Graphics graphics = Graphics.FromImage(bitmap))
    {
        graphics.SmoothingMode = SmoothingMode.HighQuality;

        // Transform to map the graph bounds to the Bitmap.
        var rect = new RectangleF(xmin, ymin, xmax - xmin, ymax - ymin);
        PointF[] points =
        {
            new PointF(0, height),
            new PointF(width, height),
            new PointF(0, 0),
        };
        graphics.Transform = new Matrix(rect, points);

        // Draw the graph.
        using (var pen = new Pen(Color.Black, 0))
        {
            // Set pen color.
            pen.Color = plotColor ?? Color.Red;

            // See how big 1 pixel is horizontally.
            var inverse = graphics.Transform;
            inverse.Invert();
            PointF[] pixel_pts =

```

```

        {
            new PointF(0, 0),
            new PointF(1, 0)
        };

inverse.TransformPoints(pixel_pts);
float dx = pixel_pts[1].X - pixel_pts[0].X;
dx /= 2;

// Loop over x values to generate points.
var graphPoints = new List<PointF>();
for (float x = xmin; x <= xmax; x += dx)
{
    bool valid_point = false;
    try
    {
        // Get the next point.
        float y = (float)function(x);

        // If the slope is reasonable, this is a valid point.
        if (graphPoints.Count == 0)
            valid_point = true;
        else
        {
            float dy = y - graphPoints[graphPoints.Count - 1].Y;
            if (Math.Abs(dy / dx) < 1000)
                valid_point = true;
        }

        if (valid_point)
            graphPoints.Add(new PointF(x, y));
    }
    catch
    {
    }

    // If the new point is invalid, draw
    // the points in the latest batch.
    if (!valid_point)
    {
        if (graphPoints.Count > 1)
            graphics.DrawLine(pen, graphPoints.ToArray());
        graphPoints.Clear();
    }
}

// Draw the last batch of points.
if (graphPoints.Count > 1)
    graphics.DrawLine(pen, graphPoints.ToArray());
    }

// Display the result.
this.picGraph.Image = bitmap;
    }
}
}

```

ПРИЛОЖЕНИЕ 5. Листинг разработанных модульных тестов

```
using System;

using FluentAssertions;
using Xunit;

using LipshMinimization.ELipschitzMath;

namespace LipshMinimizationTests
{
    public sealed class ELipschitzMathTests
    {
        private double L(double e)
            => 1.0 / (4.0 * e) + 1;

        [Theory]
        [InlineData(-2, 3, -Math.PI, 3 * Math.PI, 0.01, 0.1)]
        [InlineData(-1, 1, -Math.PI/2.0, Math.PI/2.0, 0.01, 0.02)]
        [InlineData(0, 1, 0, Math.PI, 0.05, 0.1)]
        public void TwoDimensionF1(double a, double b, double d, double c, double e, double
e2)
        {
            // Рассматриваемая функция
            double F(double x, double y)
                => Math.Abs(x) + Math.Sqrt(Math.Abs(Math.Sin(y)));

            // ожидаемый результат
            double fMin = 0;

            // полученное решение
            var result = MathStrategy.UniformSearchByBiryukov(
                F,
                a, b, // [a;b]
                d, c, // [d;c]
                L(e), // L=L(e)=1/(4e)
                e, e2); // e, e*

            // Проверка прохождения теста
            (result.F - fMin).Should().BeLessOrEqualTo(e2);
        }

        [Theory]
        [InlineData(-2, 3, -Math.PI, 3 * Math.PI, 0.01, 0.1)]
        [InlineData(-1, 1, -Math.PI / 2.0, Math.PI / 2.0, 0.01, 0.02)]
        [InlineData(0, 1, 0, Math.PI, 0.05, 0.1)]
        public void TwoDimensionF2(double a, double b, double d, double c, double e, double
e2)
        {
            // Рассматриваемая функция
            double F(double x, double y)
                => Math.Sqrt(Math.Abs(x)) + Math.Abs(Math.Sin(y));

            // ожидаемый результат
            double fMin = 0;

            // полученное решение
            var result = MathStrategy.UniformSearchByBiryukov(
                F,
                a, b, // [a;b]
                d, c, // [d;c]
                L(e), // L=L(e)=1/(4e)
                e, e2); // e, e*
```

```

        // Проверка прохождения теста
        (result.F - fMin).Should().BeLessOrEqualTo(e2);
    }

    [Theory]
    [InlineData(-Math.PI, 3 * Math.PI, 0.001, 0.01)]
    [InlineData(-Math.PI / 2.0, Math.PI / 2.0, 0.0005, 0.001)]
    [InlineData(0, Math.PI, 0.01, 0.1)]
    public void OneDimensionF1(double a, double b, double e, double e2)
    {
        // Рассматриваемая функция
        double F(double x)
            => Math.Abs(x) + Math.Sqrt(Math.Abs(Math.Sin(x)));

        // ожидаемый результат
        double fMin = 0;

        // полученное решение
        var result = MathStrategy.UniformSearchByBiryukov(
            F,
            a, b,    // [a;b]
            L(e),    // L=L(e)=1/(4e)
            e, e2); // e, e*

        // Проверка прохождения теста
        (result.F - fMin).Should().BeLessOrEqualTo(e2);
    }

    [Theory]
    [InlineData(-15, 15, -4, -1, 3, -1, -1.005, 0.5, 0.0001, 0.001)]
    [InlineData(-5, 5, -2, 1, 3, 0, -0.01, 0, 0.001, 0.01)]
    public void OneDimensionF3(double a, double b, double a1, double a2, double a3,
double b1, double b2, double b3, double e, double e2)
    {
        // Рассматриваемая функция
        double F(double x)
            => Math.Min(
                Math.Min(
                    Math.Sqrt(Math.Abs(x - a1)) + b1,
                    Math.Sqrt(Math.Abs(x - a2)) + b2),
                Math.Sqrt(Math.Abs(x - a3)) + b3);

        // ожидаемый результат
        double fMin = 0;

        // полученное решение
        var result = MathStrategy.UniformSearchByBiryukov(
            F,
            a, b,    // [a;b]
            L(e),    // L=L(e)=1/(4e)
            e, e2); // e, e*

        // Проверка прохождения теста
        (result.F - fMin).Should().BeLessOrEqualTo(e2);
    }

    [Theory]
    [InlineData(-Math.PI, 3 * Math.PI, 0.001, 0.01)]
    [InlineData(-Math.PI / 2.0, Math.PI / 2.0, 0.0005, 0.001)]
    [InlineData(0, Math.PI, 0.01, 0.1)]
    public void OneDimensionF2(double a, double b, double e, double e2)
    {
        // Рассматриваемая функция

```

```

double F(double x)
    => Math.Sqrt(Math.Abs(x)) + Math.Abs(Math.Sin(x));

// ожидаемый результат
double fMin = 0;

// полученное решение
var result = MathStrategy.UniformSearchByBiryukov(
    F,
    a, b,    // [a;b]
    L(e),    // L=L(e)=1/(4e)
    e, e2); // e, e*

// Проверка прохождения теста
(result.F - fMin).Should().BeLessOrEqualTo(e2);
}
}
}

```

ПРИЛОЖЕНИЕ 6. Презентация

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Казанский национальный исследовательский технический университет
им. А.Н. Туполева-КАИ»
КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
им. Ю.В. Кожевникова

1

Выпускная квалификационная работа на тему: Реализация численных методов решения негладких экстремальных задач

Выполнил: студент группы 4412
Бирюков А. М.

Руководитель: д.т.н., профессор
Заботин В. И.

Казань, 2019

Введение

2

Из работ Vanderbei известно, что всякая непрерывная на выпуклом компакте функция удовлетворяет условию

$$\forall \varepsilon > 0 \exists L_\varepsilon \forall x, y \in A: |f(x) - f(y)| \leq L_\varepsilon \|x - y\| + \varepsilon$$

Представляет интерес разработка приближённых алгоритмов минимизации такой функции. В работах Vanderbei и Н. К. Арутюновой предложены обобщения алгоритмов Пиявского и Евтушенко. В предлагаемой работе строятся алгоритмы, обобщающие метод равномерного перебора.

Постановка задачи для одномерного случая ³

Пусть функция $f(x)$ непрерывна на отрезке $[a; b]$, то есть удовлетворяет условию Vanderbei:

$$\forall \varepsilon > 0 \exists L_\varepsilon \forall x, y \in A: |f(x) - f(y)| \leq L_\varepsilon \|x - y\| + \varepsilon$$

Будем полагать, что оценка L_ε может быть найдена для заданного $\varepsilon > 0$. Обозначим $\varepsilon^* > \varepsilon$ точность, с которой нужно найти наименьшее значение f на отрезке $[a; b]$.

Требуется построить обобщение метода перебора на равномерной сетке, обеспечивающее отыскание наименьшего значения f с заданной точностью.

Алгоритм минимизации непрерывной функции одной переменной ⁴

Здесь и далее полагаем, что f и ε согласованы, то есть найдутся точки x и y : $|f(x) - f(y)| > \varepsilon$

1. Задать $\varepsilon > 0$ и $\varepsilon^* > \varepsilon$ – параметры метода.
2. Вычислить L_ε
3. Построить по формуле: $x_{i+1} = x_i + h, i = 0, 1, \dots, n-1$, где $h = \frac{b-a}{n}$, удовлетворяющее неравенству $h \leq \frac{\varepsilon^* - \varepsilon}{L_\varepsilon}$
Отсюда получаем число узлов $n \geq \frac{L_\varepsilon(b-a)}{\varepsilon^* - \varepsilon}$.
4. Вычисляем значения функции в полученных узлах и выбираем наименьшее из них.

Обоснование алгоритма

Обозначим:

f^* - наименьшее значение функции на отрезке (которое, очевидно, существует).

Если выполнены все предположения предыдущего слайда и мы можем вычислить оценку L_ε для любого $\varepsilon > 0$, то доказано следующее

Утверждение 1:

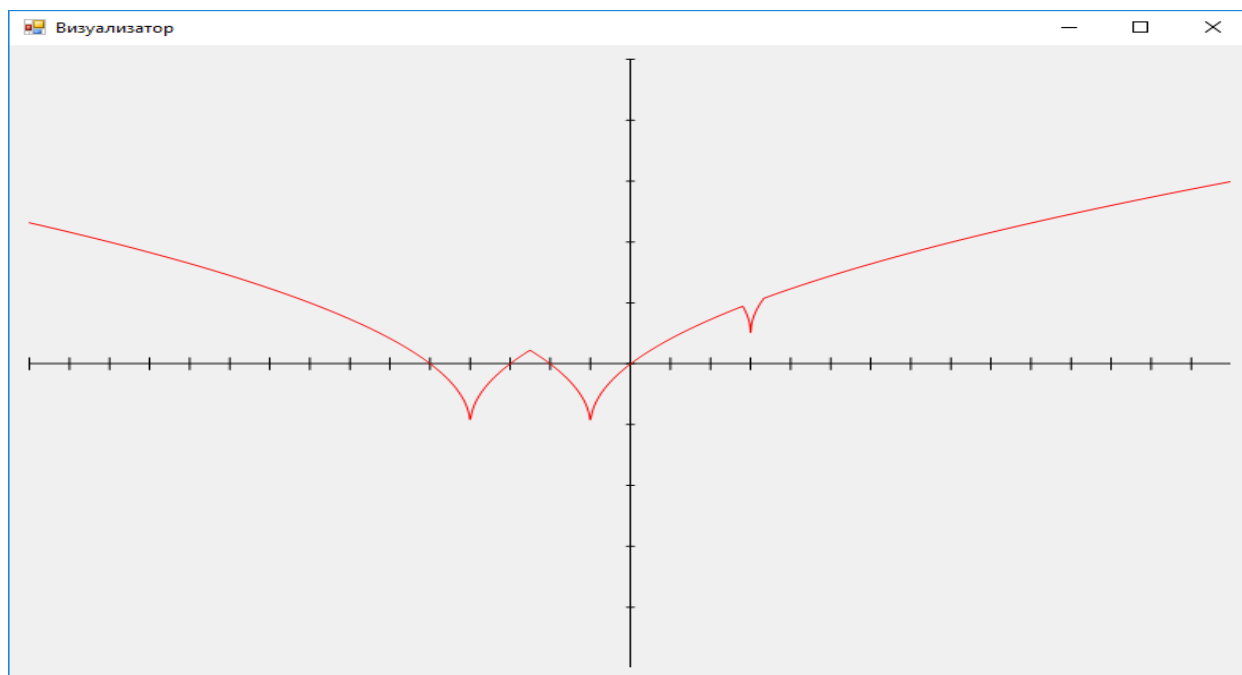
среди членов последовательности найдётся точка, для которой выполняется условие

$$f(x_k) - f^* \leq \varepsilon^*$$

Пример (Н.К. Арутюнова)

6.1

$$f(x) = \min \{ \sqrt{|x - a_1|} + b_1, \sqrt{|x - a_2|} + b_2, \sqrt{|x - a_3|} + b_3 \}$$



Пример (Н.К. Арутюнова)

Отрезок	ε^*	ε	h	$L(\varepsilon)$	x_{min}	F_n	n	$t, \text{мс}$
Набор 2. $b_1 = -1, b_2 = -1.005, b_3 = 0.5; a_1 = -4, a_2 = -1, a_3 = 3$								
[-5; 5]	0.01	0.005	50	0.000100	-0.999999	-1.004999	99999	3
		0.001	250	0.000036	-1.000004	-1.003000	277777	9
	0.001	0.0005	500	0.000001	-0.999999	-1.004982	9999999	367
		0.0001	2500	0.0000004	-1.000000	-1.004798	27777777	997
[-10; 10]	0.01	0.005	50	0.000100	-1.000000	-1.004997	200000	7
		0.001	250	0.000036	-1.000000	-1.004993	555555	23
	0.001	0.0005	500	0.000001	-1.000000	-1.004973	20000000	711
		0.0001	2500	0.0000004	-1.000000	-1.004919	55555555	1971
[-15; 15]	0.01	0.005	50	0.000100	-1.000000	-1.004995	300000	10
		0.001	250	0.000036	-0.999996	-1.003000	833333	43
	0.001	0.0005	500	0.000001	-1.000000	-1.004933	30000000	1056
		0.0001	2500	0.0000004	-1.000000	-1.004834	83333333	2871

Постановка задачи для функции двух переменных на брус

Пусть функция $f(x, y)$ непрерывна на брус $P = [a; b] \times [d; c]$, то есть удовлетворяет условию Vanderbei:

$$\forall \varepsilon > 0 \exists L_\varepsilon \forall x, y \in P: |f(x) - f(y)| \leq L_\varepsilon \|x - y\| + \varepsilon$$

Будем полагать, что оценка L_ε может быть найдена для заданного $\varepsilon > 0$. Обозначим $\varepsilon^* > \varepsilon$ точность, с которой нужно найти наименьшее значение f на брус P .

Требуется построить обобщение метода перебора на равномерной сетке, обеспечивающее отыскание наименьшего значения f с заданной точностью.

Алгоритм минимизации непрерывной функции двух переменных 8

Полагаем, что f и ε согласованы, то есть найдутся точки $(x_1, y_1), (x_2, y_2) \in P : |f(x_1, y_1) - f(x_2, y_2)| > \varepsilon$

1. Задать $\varepsilon > 0$ и $\varepsilon^* > \varepsilon$, вычислить L_ε .
2. Найти $m \geq \frac{(c-d)L_\varepsilon}{\varepsilon^* - \varepsilon}$ и $n \geq \frac{(b-a)L_\varepsilon}{\varepsilon^* - \varepsilon}$.
3. Найти узлы сетки $(x_i, y_j), i = \overline{0, n-1}, j = \overline{0, m-1}$ с шагом $h_x = \frac{b-a}{n}$ и $h_y = \frac{c-d}{m}$.
4. Вычислить значения функции в узлах и выбрать наименьшее.

Обоснование алгоритма 9

Обозначим:

f^* - наименьшее значение функции на бруске P (которое, очевидно, существует).

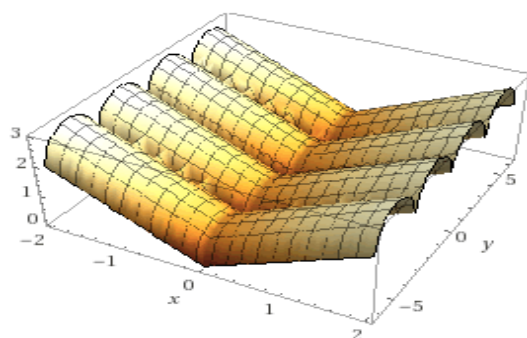
Если выполнены все предположения предыдущего слайда и мы можем вычислить оценку L_ε для любого $\varepsilon > 0$, то доказано следующее

Утверждение 2:

среди членов последовательности найдётся точка (x_i, y_j) , для которой выполняется условие

$$f(x_i, y_j) \leq f^* + \varepsilon^*$$

Пример



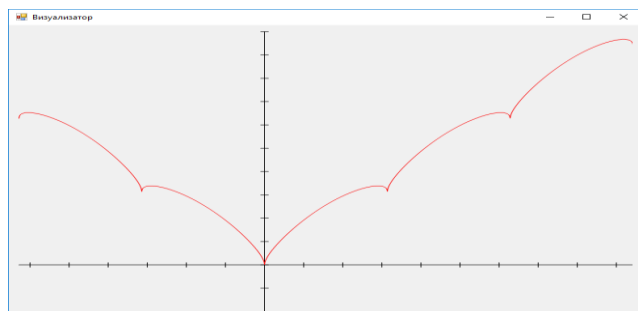
$$f(x, y) = |x| + \sqrt{|\sin y|}$$

Брус	ϵ^*	ϵ	$L(\epsilon)$	h	x_{min}	y_{min}	F_n	n	m	$t, \text{мс}$
$[-1; 1]$ \times $[-\frac{\pi}{2}; \frac{\pi}{2}]$	0.05	0.01	26	0.001538	0.000000	-0.000027	0.005210	1300	2043	117
		0.001	251	0.000195	-0.000088	-0.000063	0.008044	10245	16093	7191
	0.01	0.005	51	0.000098	0.000000	-0.000012	0.003472	20400	32045	28118
		0.001	251	0.000036	0.000004	0.000008	0.002904	55778	87616	221548
$[-0.5; 0.5]$ \times $[0; \pi]$	0.05	0.01	26	0.001538	0.000000	0.000000	0.000000	650	2043	65
		0.001	251	0.000195	-0.000044	0.000000	0.000044	5123	16093	3627
	0.01	0.005	51	0.000098	0.000000	0.000000	0.000000	10200	32045	14249
		0.001	251	0.000036	-0.000016	0.000000	0.000016	27889	87616	106840

11.1

Расчёт тестовых примеров

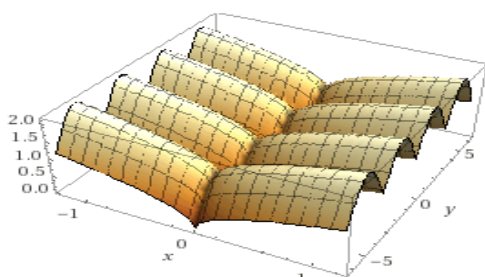
$$f(x) = |x| + \sqrt{|\sin x|}$$



Отрезок	ϵ^*	ϵ	$L(\epsilon)$	h	x_{min}	F_n	n	$t, \text{мс}$
$[-2\pi; 3\pi]$	0.01	0.005	51	0.000098	0.000044	0.006647	2796	0
		0.001	251	0.000036	-0.000013	0.0036	7645	0
	0.001	0.0005	501	0.000001	0.000000	0.000636	274704	14
		0.0001	2501	0.0000004	0.000000	0.00031	761848	32

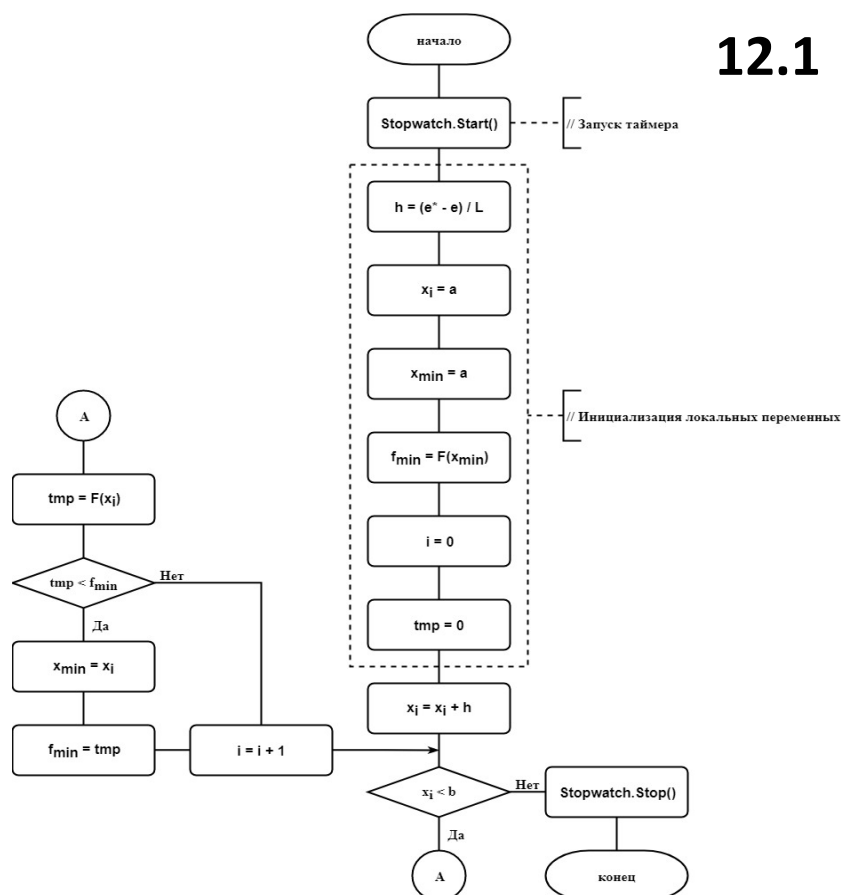
Расчёт тестовых примеров

$$f(x, y) = \sqrt{|x|} + |\sin y|$$

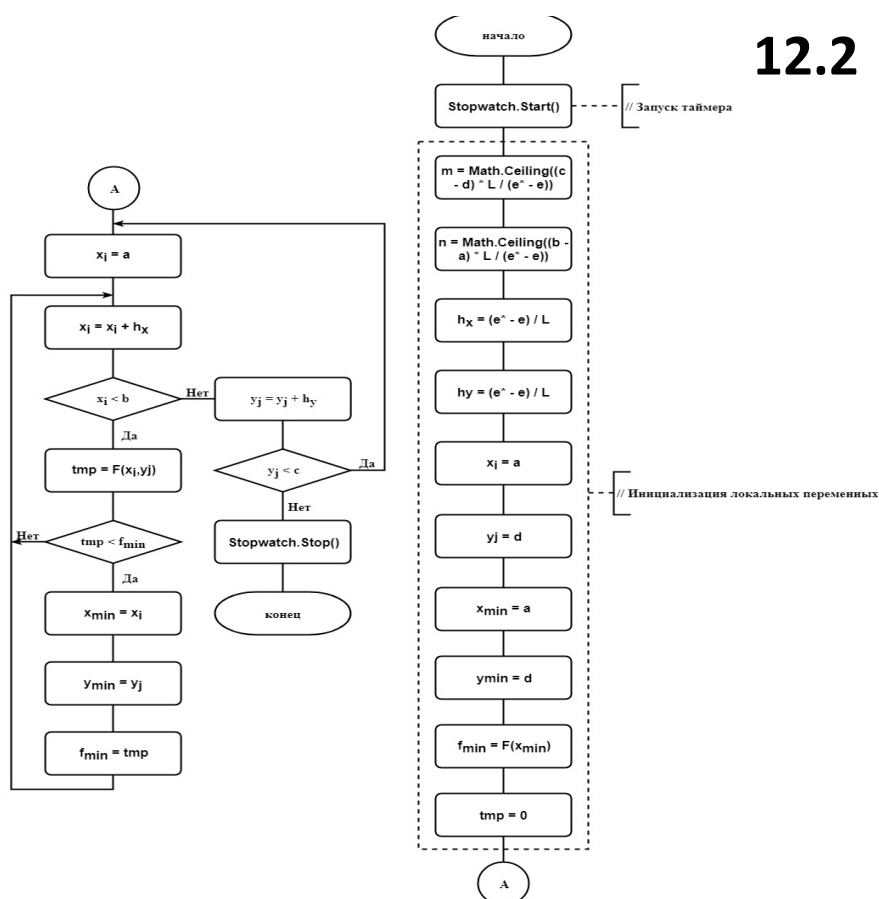


Брус	ε^*	ε	$L(\varepsilon)$	h	x_{min}	y_{min}	F_n	n	m	t, MC
$[-1; 1]$ \times $[-\frac{\pi}{2}; \frac{\pi}{2}]$	0.05	0.01	26	0.001538	0.000000	-0.000027	0.000027	1300	2043	122
		0.001	251	0.000195	-0.000088	-0.000063	0.009425	10245	16093	7390
	0.01	0.005	51	0.000098	0.000000	-0.000012	0.000012	20400	32045	29085
		0.001	251	0.000036	0.000004	0.000008	0.002004	55778	87616	217442

Структурная
схема
алгоритма
для
непрерывной
функции
одной
переменной



Структурная схема алгоритма для функции двух переменных на брус



Заключение

13

- Изучено современное состояние методов приближённого вычисления минимального значения ε -липшицевых на выпуклом компакте функций;
- разработаны, описаны и обоснованы следующие численные методы:
 - алгоритм приближённого поиска глобального минимума непрерывной на отрезке функции, основанный на идеях равномерного перебора;
 - алгоритм приближённого поиска глобального минимума непрерывной на брус функции двух переменных, также основанный на идее равномерного перебора по сетке.


Следует подчеркнуть, что в обоих случаях на функции не накладывались никакие дополнительные требования, кроме их непрерывности на выпуклом компакте конечномерного арифметического пространства.

- разработано программное обеспечение для обоих алгоритмов;
- с помощью разработанного программного обеспечения рассчитана серия тестовых задач. Для некоторых ранее известных результатов поиска глобального минимума проведено сравнение с полученными предложенными методами, которое показало хорошее совпадение приближённых минимальных значений функции.

Список использованной литературы 14

1. Vanderbei, R.J. Extension of Piyavskii's Algorithm to Continuous Global Optimization / R.J. Vanderbei // Journal of Global Optimization. – 1999. – Vol. 14. – P. 205-216.
2. Заботин, В.И. Алгоритм вычисления минимальной оценки ε -постоянной Липшица непрерывной функции / В.И. Заботин, П.А. Чернышевский // Вестник КГТУ им. А.Н. Туполева. – 2018. - № 2, вып. 2. – С. 127-132.
3. Заботин, В.И. Два алгоритма отыскания проекции точки на невыпуклое множество в нормированном пространстве / В.И. Заботин, Н.К. Арутюнова // Журнал вычислительной математики и математической физики. – 2013. – Т. 53, № 3. – С. 344-349.
4. Арутюнова, Н.К. Алгоритмы проектирования точки на поверхность уровня непрерывной на компакте функции / Н.К. Арутюнова, А.М. Дуллив, В.И. Заботин // Журнал вычислительной математики и математической физики. – 2014. – Т. 54, № 9. – С. 1448-1454.
5. Арутюнова, Н.К. Модификация метода Евтушенко поиска глобального минимума для случая непрерывной на компакте функции / Н.К. Арутюнова // Вестник КГТУ им. А.Н. Туполева. – 2013. - № 2, вып. 2. – С. 154-157.
6. N.K. Arutyunova Models and methods for three external ballistics inverse problems / N.K. Arutyunova, A.M. Dulliev, V.I. Zabotin // , Vestnik YuUrGU. – 2017. – Vol. 10. – P. 78-91.
7. ГОСТ 19.701–90. Единая система программной документации. – Москва: Стандартинформ, 2010. – 22 с.
8. Технологии разработки программного обеспечения. Глава 6. Структурное тестирование программного обеспечения [Электронный ресурс] – Режим доступа : <http://textarchive.ru/c-1144105-p14.html>.
9. [Technology Conversations](https://technologyconversations.com/2013/12/20/test-driven-development-tdd-example-walkthrough/). Test Driven Development (TDD): Example Walkthrough. [Электронный ресурс] – Режим доступа : <https://technologyconversations.com/2013/12/20/test-driven-development-tdd-example-walkthrough/>
10. xUnit.net. About xUnit.net. [Электронный ресурс] – Режим доступа : <https://xunit.net/>


Публикация 15



Всероссийская молодёжная научно-практическая конференция

УНИИД ПГУ им. Шолом-Алейхема <uniid@bk.ru>

Контакт: aleks-herou@mail.ru

6 мая, 2:13  1 файл

Добрый день!

Ваша статья прошла рецензирование и принята к публикации в сборнике трудов конференции.

Так же направляю электронный вариант сертификата участника.

