ФГБОУ ВПО «Липецкий государственный технический университет»

Кафедра прикладной математики

Учебно-исследовательская работа

по дисциплине: «Алгоритмы оптимизации»

Выполнили: студенты гр. ПМ-08-2
Лукьянчиков В.С.
Петрухина Е.В.
Сотников Р.А.
Боева М.В.

Липецк - 2012

- **Проект:**

  Исследование алгоритмов глобальной оптимизации

- **Цель:**

  Реализация и исследование качества работы и эффективности алгоритмов глобальной оптимизации функций

- **Задание:**

1. Разработать программное обеспечение для глобальной оптимизации функций на основе методов Монте-Карло, имитации обжига, генетических алгоритмов, интервальных методов.
2. Провести исследования и сравнительный анализ качества и эффективности работы алгоритмов на нескольких (не менее трёх) тестовых задачах.
3. Выявить параметры, наиболее сильно влияющие на качество и эффективность алгоритмов глобальной оптимизации.
4. Сделать выводы о работе на основе результатов исследования и сравнительного анализа алгоритмов глобальной оптимизации, указать возможные способы усовершенствования алгоритмов.

Метод Монте-Карло

- Заключается в генерировании бесконечно большого количества случайных точек, в каждой из которых вычисляется значение целевой функции. Результат работы - точка, которая приводит к наименьшему значению функции.
- Метод Монте-Карло - базовый алгоритм стохастической оптимизации
- ***Алгоритм 1.***

  Инициализация:

  $f_{min:}=\infty$, $x_{min}=NaN$ (Not A Number – неопределенность типа (0/0)),
  N – очень большое целое число – число генерируемых точек, $i:=0$,

  1. Цикл: повторять пока $i < N$

     1.1. x_i := случайная точка

     1.2. Если $f(x_i)<f_{min}$, то $x_{min}=x_i$, $f_{min} =f(x_i)$

     1.3. i:=i+1 и перейти на шаг 1.1.
- Если значения функции вычисляются в точках, полученных на основе равномерного распределения области *S*, наименьшее значение функции сходится к глобальному минимуму с вероятностью 1.

Программная реализация метода Монте-Карло

Шаг 1: вводим количество переменных исследуемой функции

Шаг 2: указываем исследуемую функцию

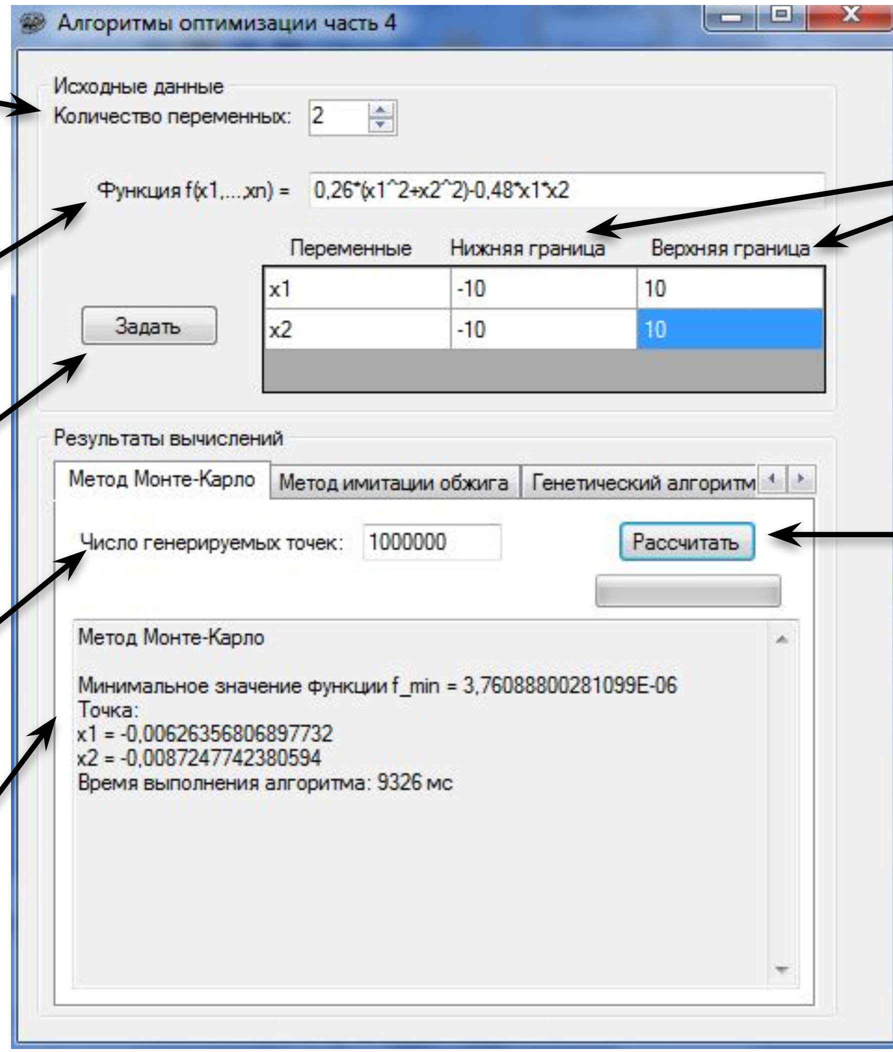
Шаг 3: указываем границы отрезка, на котором производится поиск оптимума

Шаг 4: подтверждаем введенные данные, нажав кнопку «Задать»

Шаг 5: указываем количество точек, среди которых будет проводиться поиск минимума

Шаг 6: приступаем к поиску решения, нажав кнопку «Рассчитать»

Шаг 7: непосредственно ответ



Метод имитации обжига

- Алгоритм имитации обжига отражает поведение расплавленного материала при отвердевании с применением процедуры отжига (управляемого охлаждения) при температуре, последовательно понижаемой до нуля.
- В процессе медленного управляемого охлаждения, называемого обжигом, кристаллизация расплава сопровождается глобальным уменьшением его энергии, однако допускаются ситуации, в которых она может на какое-то время возрастать.
- ***Алгоритм 2.***

Инициализация:

$T := T_{max} > 0$ – максимальная температура (большое вещественное число)

L – количество циклов для каждой температуры (целое число)

r из интервала (0;1) – параметр снижения температуры (вещественное число)

$eps > 0$ – малое вещественное число (например, 1e-10)

1. Выбрать случайную точку x
2. Пока $T > 0$ повторять L раз следующие действия:
   2.1. Выбрать новую точку x′ из *eps*-окрестности точки x
   2.2. Рассчитать изменение целевой функции $\Delta = f(x') - f(x)$
   Если $\Delta <= 0$, то $x := x'$ иначе
   Если $e^{-\Delta/T} >$ случайного числа, р/р на интервале (0;1), то $x := x'$
3. Уменьшить температуру $T := rT$. Вернуться к пункту 2.
4. Провести оптимизацию любым методом локальной оптимизации.

Программная реализация метода имитации обжига

Шаг 1: вводим количество переменных исследуемой функции

Шаг 2: указываем исследуемую функцию

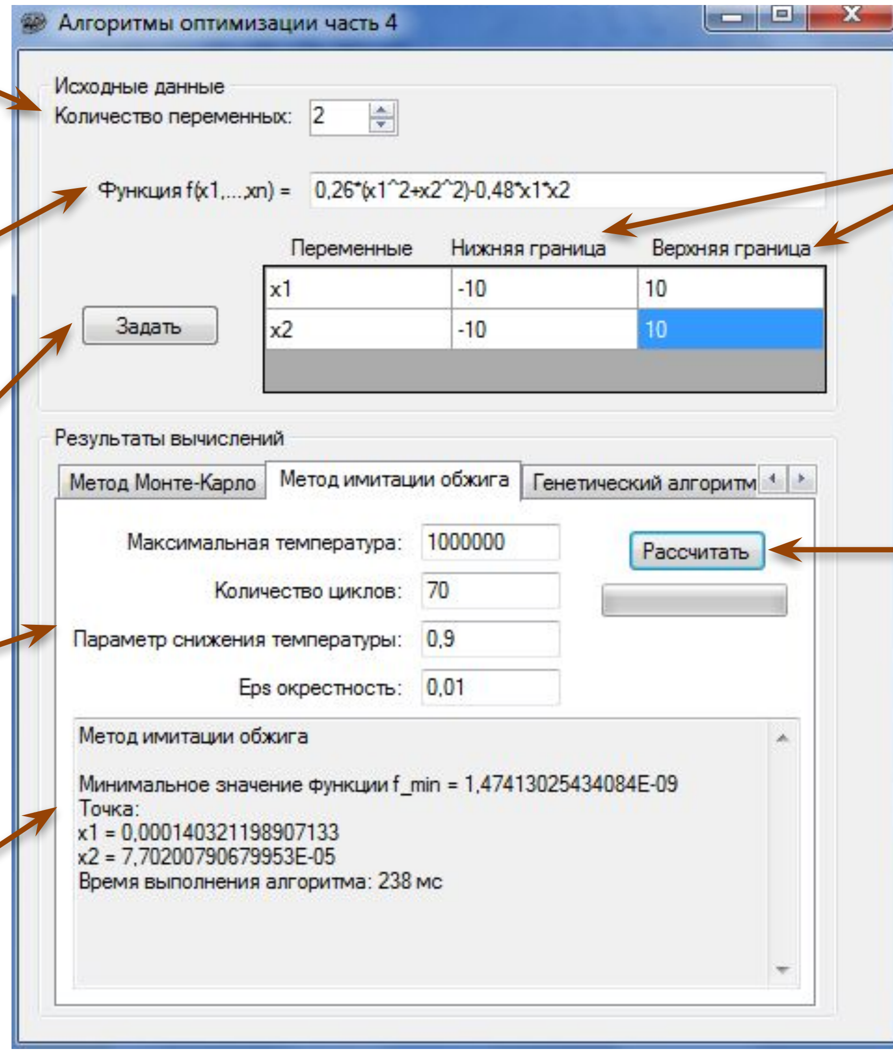
Шаг 3: указываем границы отрезка, на котором производится поиск оптимума

Шаг 4: подтверждаем введенные данные, нажав кнопку «Задать»

Шаг 5: указываем необходимые параметры

Шаг 6: приступаем к поиску решения, нажав кнопку «Рассчитать»

Шаг 7: и получаем ответ



Генетические алгоритмы

Генетические алгоритмы – смена поколений на основе операторов отбора, скрещивания, мутации, редукции.

Основные понятия ГА:

- **Фитнесс-функция:** f(x).
- **Особь (**хромосома, индивид**):** $x = (x_1x_2x_3\ldots x_n)$,
- **Ген** - бит строки x_i.
- **Популяция** - $X = \{x_i,\ i=1,\ldots,k\}$.
- Работа ГА - **смена поколений**.

Алгоритм 3.

1. Создание исходной популяции.
2. Выбор родителей для процесса размножения (оператор отбора).
3. Создание потомков выбранных пар родителей (оператор скрещивания).
4. Мутация новых особей (оператор мутации).
5. Сокращение расширенной популяции до исходного размера (оператор редукции).
6. Проверка выполнения критерия останова. Если не выполнен, то переход на шаг 2.
7. Выбор лучшей достигнутой особи в конечной популяции в качестве решения.

Программная реализация генетического алгоритма оптимизации

Шаг 1: вводим количество переменных исследуемой функции

Шаг 2: указываем исследуемую функцию

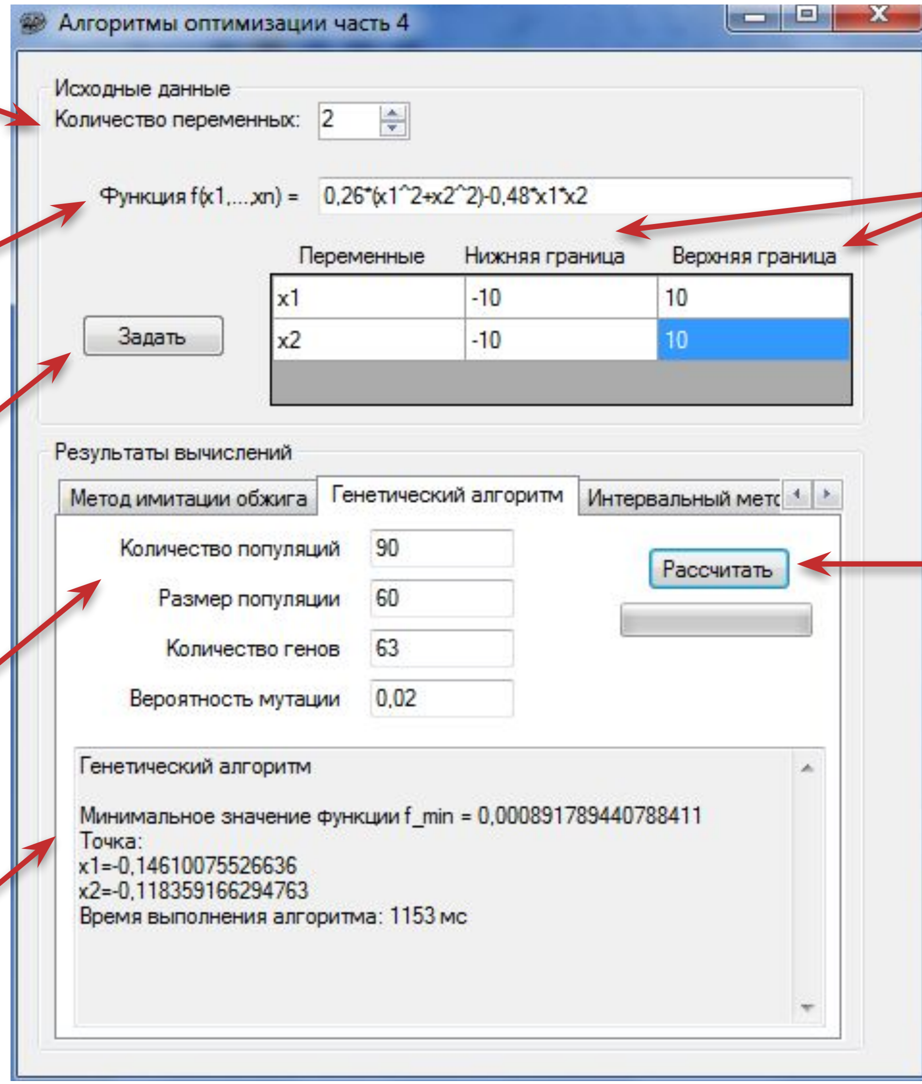
Шаг 3: указываем границы отрезка, на котором производится поиск оптимума

Шаг 4: подтверждаем введенные данные, нажав кнопку «Задать»

Шаг 5: указываем необходимые параметры

Шаг 6: приступаем к поиску решения, нажав кнопку «Рассчитать»

Шаг 7: и перед нами ответ



Интервальный анализ

- Интервальная арифметика – расширение арифметики действительных чисел на случай интервалов.
- Основы интервального анализа:

X, Y, Z – множества, $* : X \times Y \to Z$ – бинарное отображение.

Расширение на множества:

$$X_1 * Y_1 = \{x * y \mid x \in X_1 \subset X, y \in Y_1 \subset Y\}$$

Если $X_1, Y_1 \subset R^n$

то

$$X_1 + Y_1 = \{x + y \mid x \in X_1 \subset R^n, y \in Y_1 \subset R^n\}$$

$$X_1 - Y_1 = \{x - y \mid x \in X_1 \subset R^n, y \in Y_1 \subset R^n\}$$

- Основа интервальных алгоритмов глобальной оптимизации– итерационная процедура разбиения исходного бруса на подбрусы (бисекция) и исследование поведения функции на каждом подбрусе.
- Для отсеивания неперспективных брусов используются тесты в средней точке, на монотонность, на выпуклость.
- С помощью интервального анализа возможно нахождение всех глобальных оптимумов.

Алгоритм 4 (алгоритм поиска всех глобальных оптимумов).

Вход: Функция $f(x), x \in R^n$; $f'(x), f''(x)$, $[x]$ – начальный брус; минимальная ширина бруса $\varepsilon > 0$.

Выход: L_{res} – список брусов, содержащих точки глобального минимума; $[f^*]$ – оценка глобального минимума.

1. Инициализация: $[p] := [x]$, $c :=$ mid($[p]$)

2. Оценка верхней границы минимума:

3. Инициализация списков: $L := \{\}$, $L_{res} := \{\}$

4. Главный ЦИКЛ:

4.1.Выбираем компоненту l, по которой брус $[p]$ имеет наибольшую длину: $l :=$ arg max wid($[p_i]$)

4.2. Бисекция $[p]$ по l-й координате на $[p_1]$ и $[p_2]$

4.3. Цикл по $i := 1..2$

4.3.1. $[g] := [f']([p_i])$ – функция включения для градиента

4.3.2. Если тест на монотонность не пройден, то переход на следующий i

4.3.3. $[f]_c := (f(m)+[g]([p_i]-m))$ – центрированная форма функции включения

4.3.4. Если тест на нижнюю границу не пройден, т.е. $\tilde{f} < \underline{[f]_c}$, то переход на следующий i

4.3.5. $[H]$:= $[f'](\[p_i])$ – функция включения для матрицы Гессе

4.3.6. Если тест на выпуклость не пройден (на главной диагонали $[H]$ есть элементы, меньшие 0) , то переход на следующий i

4.3.7. $L := L + \left(p_i, \underline{[f]([p_i])}\right)$ – сохранение в списке

4.4. ВыполнятьБисекцию := Ложь

4.5. Цикл: Пока (L <> {}) и (не ВыполнятьБисекцию)

4.5.1. := 1-й элемент списка L;

$L := L - (p, \underline{f})$ – удаление из списка; m := mid($[p]$)

4.5.2. $\tilde{f} := \min\{\tilde{f}, f(m)\}$

Удаление всех брусов из L, не проходящих тест на среднюю точку с $\tilde{f}$.

4.5.3. $[f^*] := [\underline{f}, \tilde{f}]$.

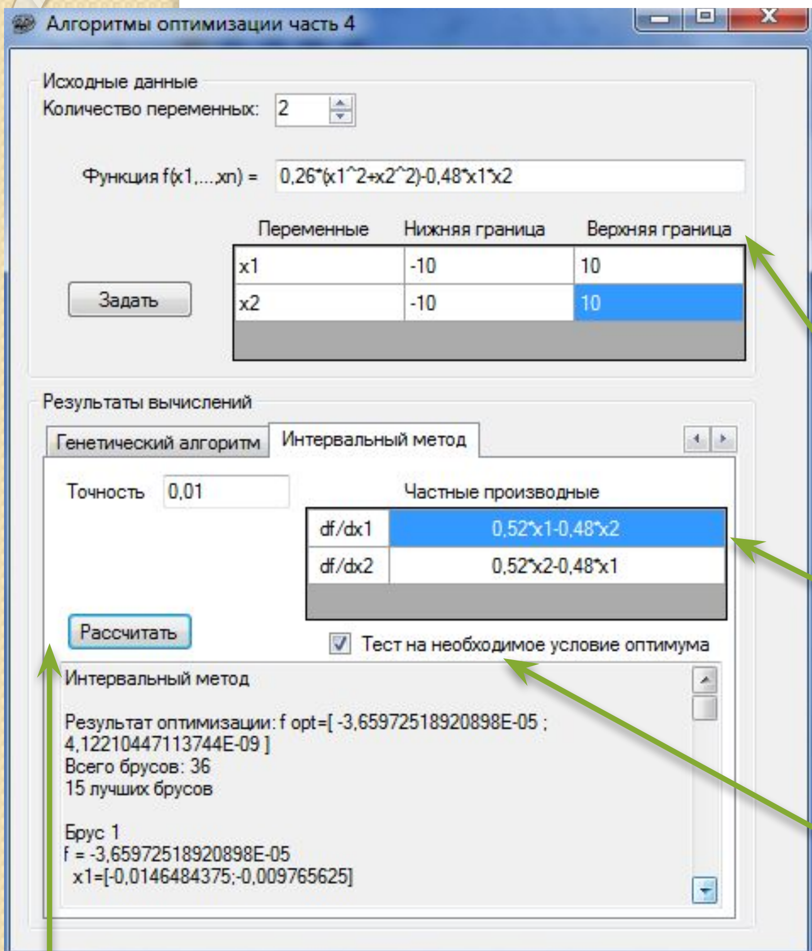
4.5.4. Если (wid($[f^*]$) ≤ ε) или (wid($[p]$) ≤ ε), то $L_{res} := L_{res} + ([p], \underline{f})$

иначе ВыполнятьБисекцию := Истина

ПОКА (ВыполнятьБисекцию)

5. $(p, \underline{f})$:= 1-й элемент списка L $[f^*] := [\underline{f}, \tilde{f}]$;

Программная реализация интервальных методов оптимизации



Шаг 1: вводим количество переменных исследуемой функции

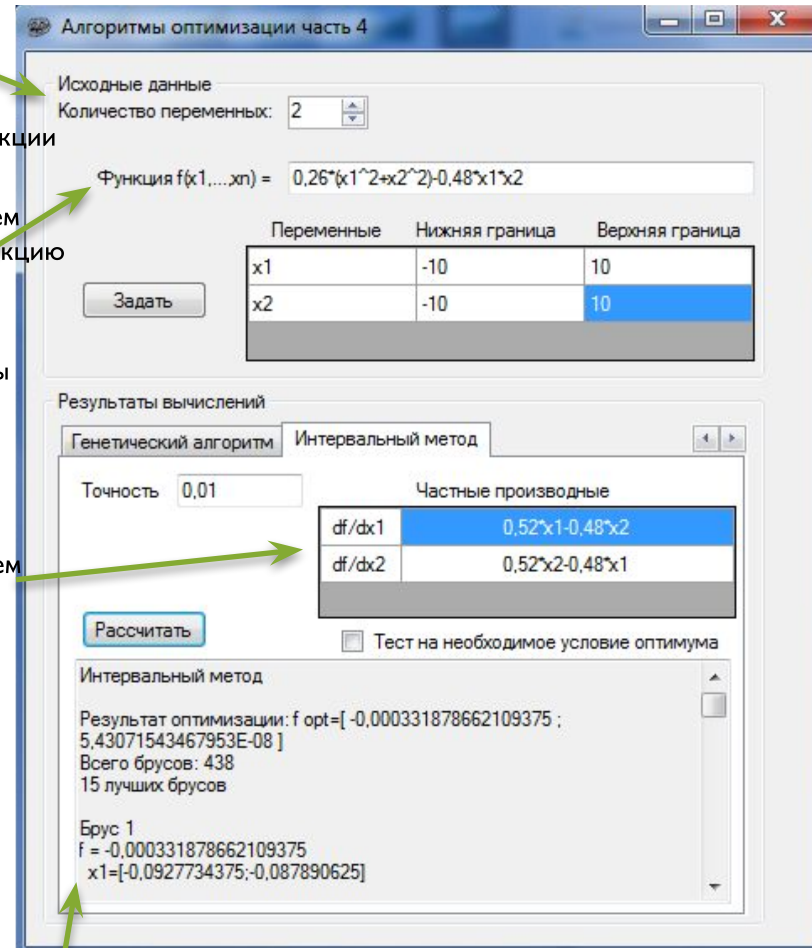
Шаг 2: указываем исследуемую функцию

Шаг 3: указываем границы отрезка, на котором производится поиск оптимума

Шаг 4: указываем первые частные производные исследуемой функции и точность вычислений

Шаг 5: указываем, нужно ли выполнять тест на необходимое условие оптимума

Шаг 6: приступаем к поиску решения, нажав кнопку «Рассчитать»



Шаг 7: и перед нами ответ

Сравнительная таблица эффективности алгоритмов оптимизации

№	Целевая функция	Результат оптимизации		
		Имитация обжига	Генетические алгоритмы	Интервальный метод
1.	(x1+2*x2-7)^2+(2*x1+x2-5)^2	f_min = 1,25093674212796E-07 x1 = 0,999749861471247 x2 = 3,00025004871201 Время:151 мс	f_min = 0,0229103771633356 x1=1,08240802539513 x2=2,88784279255196 Время:185 мс	f_min = [0 ; 4,91599415372029E-06] x1=[0,9912109375;0,99609375] x2=[2,998046875;3,0078125] Время:50 мс
2.	cos(2*x-Pi/2)^3+2*sin(x/2)	f_min = -2,85364689454974 x1 = 8,67018927781848 Время:127 мс	f_min = -2,85245745516575 x1=-3,88222089735791 Время:77 мс	f_min = [-2,87070932113906 ; -2,85364684815031] x1=[-3,896484375;-3,8916015625] Время:23 мс
3.	x1^2+x2^2-cos(12*x1)-cos(18*x2)	f_min = -1,87890065057513 x1 = -1,07175204922612E-06 x2 = 0,346921501880007 Время:147 мс	f_min = -1,9991611823282 x1=0,00318215065635741 x2=0,000782099785283208 Время:318 мс	f_min = [-2 ; -1,99998672922021] x1=[-0,00390625;0] x2=[-0,0078125;0] Время:31 мс
4.	x1^4+4*x1^3+4*x1^2+x2^2	f_min = 2,24976987496719E-08 x1 = -1,99997674716171 x2 = -0,000142600736627997 Время:159 мс	f_min = 0,0303822682796714 x1=-0,0345019402448088 x2=-0,160572718596086 Время:314 мс	f_min = [-0,000702286557441312 ; 7,30933068608796E-08] x1=[-2,0068359375;-2,001953125] x2=[-0,009765625;0] Время:146 мс
5.	0,26*(x1^2+x2^2)-0,48*x1*x2	f_min = 1,00029766810055E-09 x1 = 0,000161163346028777 x2 = 0,000151012637731052 Время:147 мс	f_min = 0,00230165580834722 x1=0,244589617941529 x2=0,224099864717573 Время:307 мс	f_min = [-3,65972518920898E-05 ; 3,36955184855087E-08] x1=[-0,0146484375;-0,009765625] x2=[-0,009765625;0] Время:80 мс

Оптимальные параметры для методов оптимизации

Метод Монте-Карло	Метод имитации обжига	Генетический алгоритм	Интервальный анализ
1. Количество генерируемых точек = 1 000 000	1. t_{max}= 100 000 2. Количество циклов =70 3. Параметр снижения температуры = 0,9 4. Eps-окрестность = 0,01	1. Количество популяций =90 2. Размер популяций = 60 3. Количество генов = 63 4. Вероятность мутации = 0,02	1. Использование теста на необходимое условие 2. Точность вычислений = 0,01

Заключение

- Были исследованы основные особенности схемы алгоритмов, тесты на проверку, особенности определения парадигм интервального анализа, а также вопросы их программной реализации, что позволило создать программный продукт с дружественным интерфейсом для оптимизации функции нескольких переменных.
- В работе произведен ряд улучшений классической схемы глобальной оптимизации, создан класс для работы с интервальным исчислением.
- В большинстве случаев алгоритмы интервального анализа более точны, нежели остальные алгоритмы.
- Время работы алгоритма интервального анализа по сравнению с другими – быстрое, реализация алгоритма значительно быстрее стохастических и генетических алгоритмов оптимизации, временные задержки могут быть связаны с особенностью интервального исчисления и переопределения функций. Разница в точности между алгоритмом с применением теста на НУ оптимума и без него практически отсутствует, однако, количество брусов, получающихся при отключении теста значительно больше, чем при его наличии, что, в свою очередь, существенно влияет на время выполнения программы.

Благодарим за внимание!