ФГБОУ ВПО «Липецкий государственный технический университет»

Кафедра прикладной математики

Учебно-исследовательская работа
по дисциплине: «Алгоритмы оптимизации»

Выполнили: студенты гр. ПМ-09-2
Литаврина Е.В.
Ляпин В.В.
Седых И.И.
Стуров А. А.
Перцев Р.А.
Шипулин Р.О.

Липецк - 2013

- **Проект:**

  Исследование алгоритмов глобальной оптимизации

- **Цель:**

  Реализация и исследование качества работы и эффективности алгоритмов глобальной оптимизации функций нескольких переменных

- **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

- Метод Монте-Карло - базовый алгоритм стохастической оптимизации
- 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.
- *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

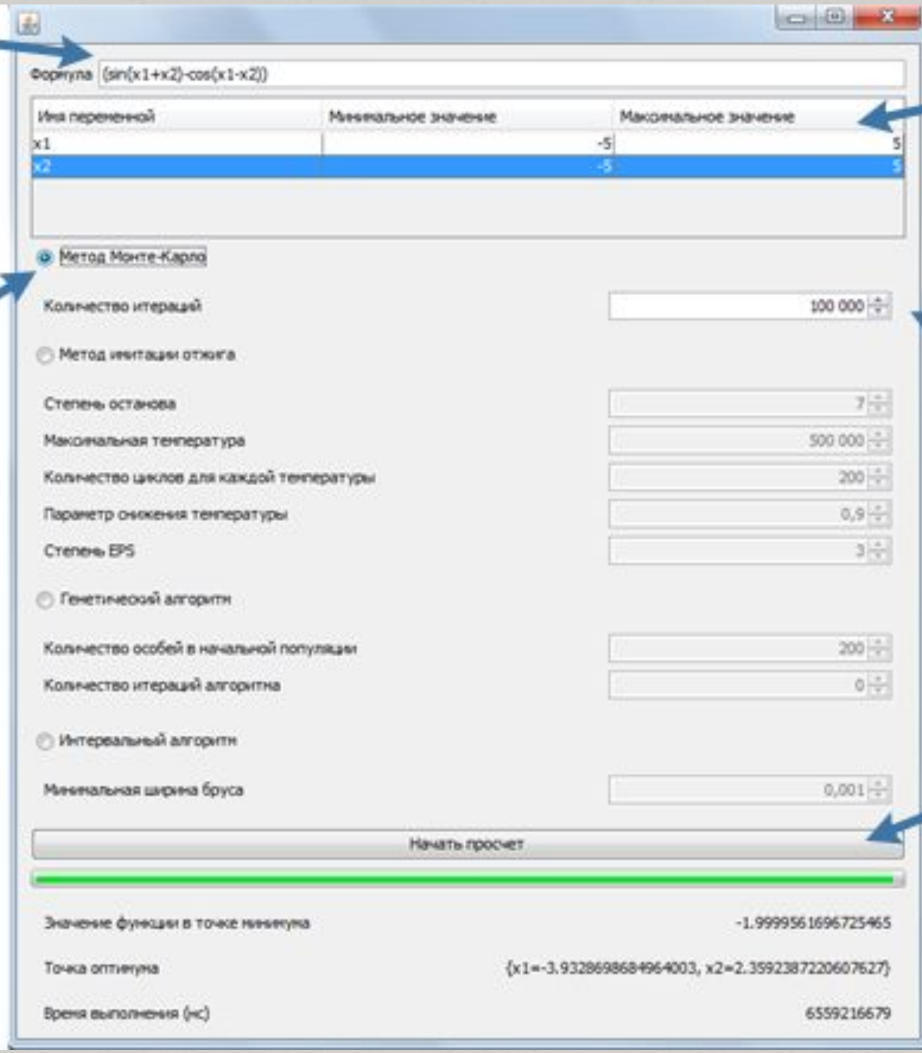
Шаг 1: Вводим целевую функцию

Шаг 2: Задаём минимальное и максимальное значения переменных

Шаг 3: Выбираем «Метод Монте Карло»

Шаг 4: Задаём количество итераций

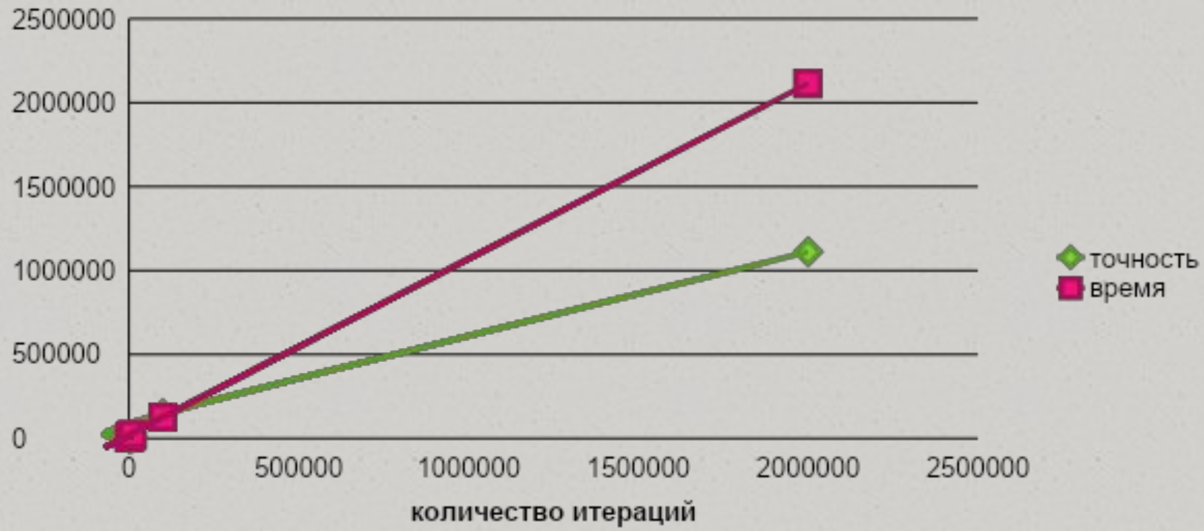
Шаг 5: Запускаем процесс вычисления



Сравнительный анализ скорости и точности метода Монте-Карло

Число генерируемых точек	Функция	Минимум функции	X_1	X_2	Время выполнения метода (нс)
1000		-1,4878408	-1,5227958	0,4639619	328396943
5000		-1,4989183	-1,5023528	0,4968789	1276326871
10000		-1,4995258	-1,4790195	0,5007679	2377426771
100000		-1,4999932	-1,4973097	0,499912	12680595740
2000000		-1,4999991	-1,499992	0,5005327	211499245441

Сравнительный анализ скорости и точности метода Монте-Карло в зависимости от количества итераций



Вывод: с увеличением числа итераций – увеличивается точность метода Монте-Карло, однако, растёт и время выполнения программы.

Оптимальным значением числа итераций можно считать 100000, так как в этом случае наиболее приемлемое отношение между точностью вычислений и скоростью вычислений.

Метод имитации обжига

- Экзотическое название данного алгоритма связано с методами имитационного моделирования в статистической физике, основанными на технике Монте-Карло. Алгоритм имитации обжига отражает поведение расплавленного материала при отвердевании с применением процедуры отжига (управляемого охлаждения) при температуре, последовательно понижаемой до нуля.
- Сегодня этот алгоритм является популярным как среди практиков благодаря своей простоте, гибкости и эффективности, так и среди теоретиков, поскольку для данного алгоритма удается аналитически исследовать его свойства и доказать асимптотическую сходимость.
- ***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 Δ=$f(x')-f(x)$

   Если Δ<=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

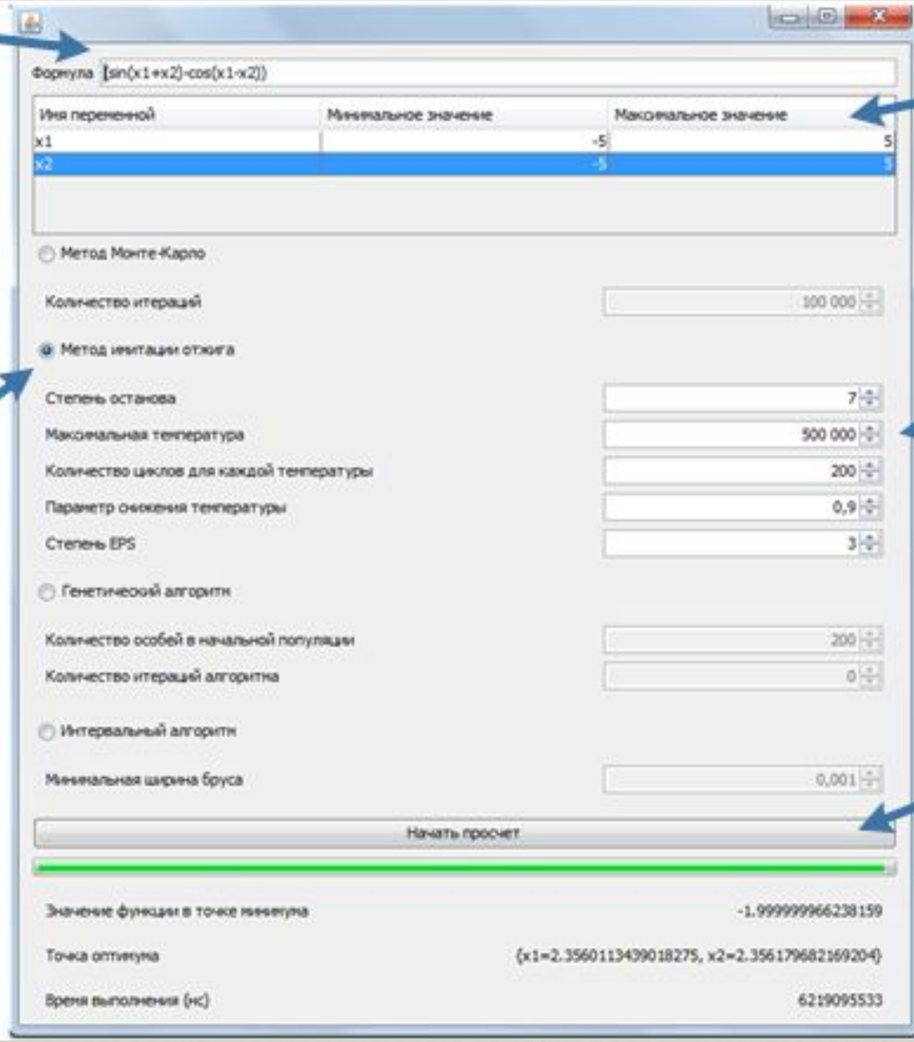
Шаг 1: Вводим целевую функцию

Шаг 2: Задаём минимальное и максимальное значения переменных

Шаг 3: Выбираем «Метод Имитации обжига»

Шаг 4: Задаём степень останова, максимальную температуру, количество циклов для каждой температуры, параметр снижения температуры и степень EPS

Шаг 5: Запускаем процесс вычисления

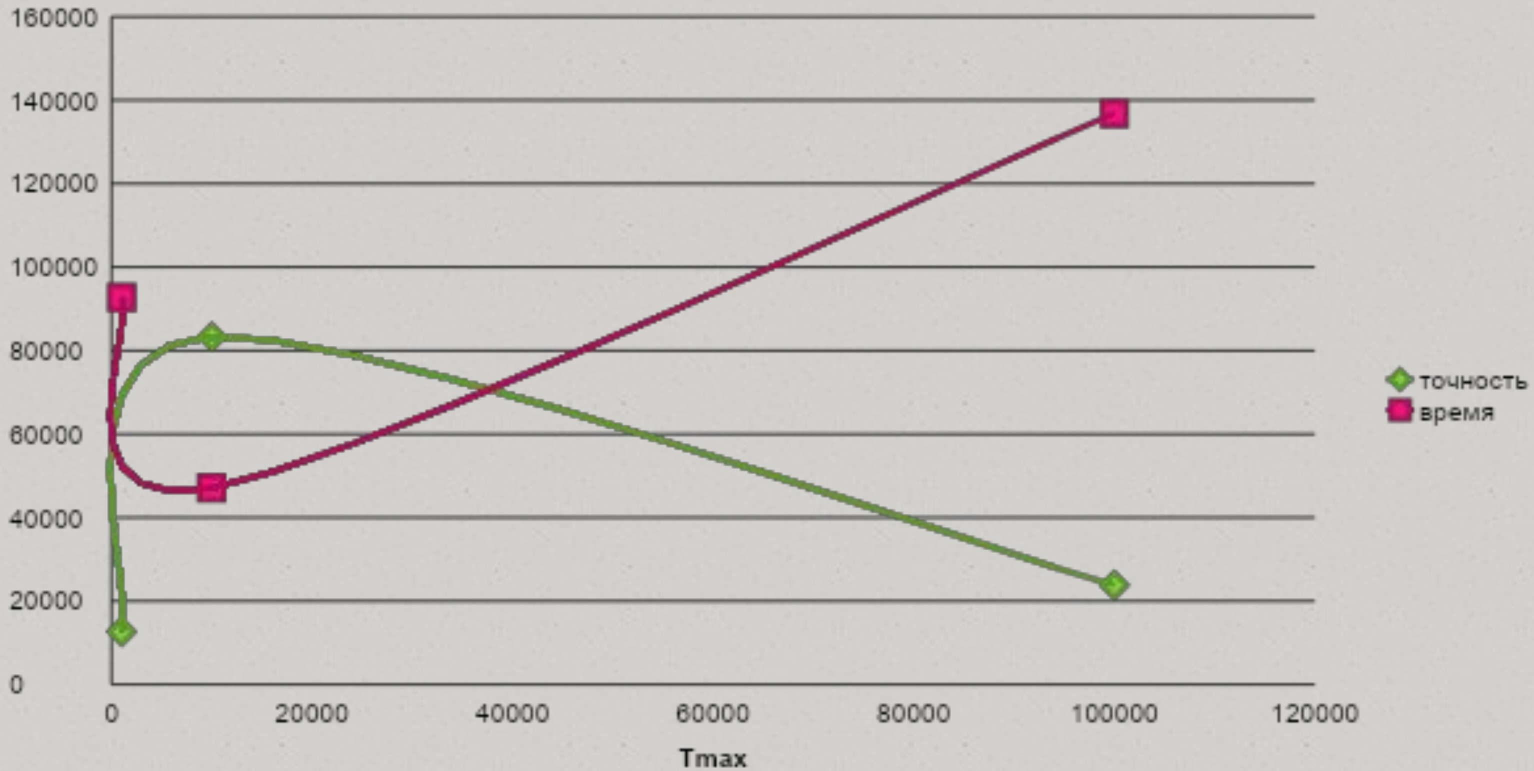


Сравнительный анализ скорости и точности метода имитации обжига

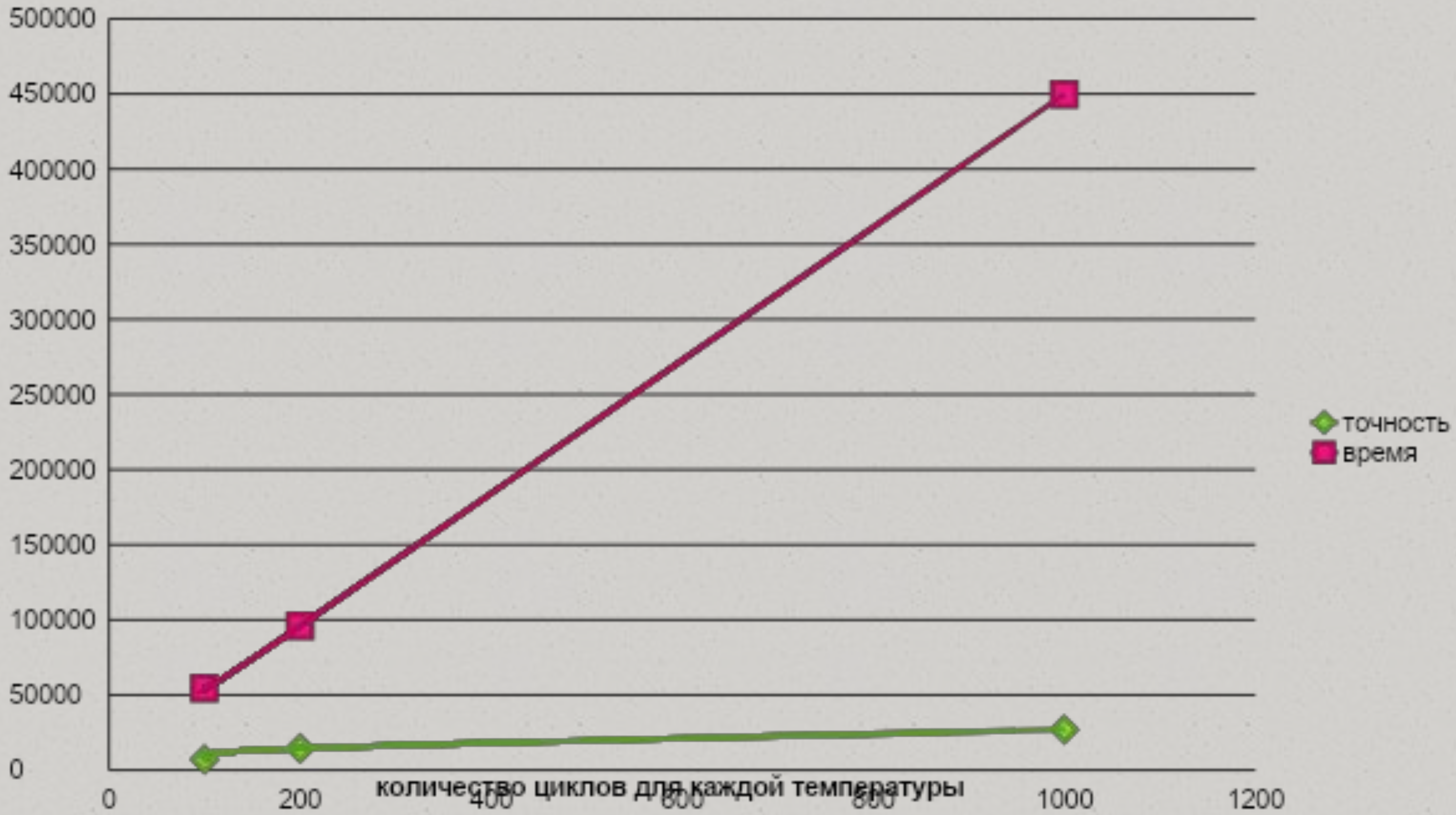
Параметр снижения T (T=50000; L=200;EPS=0,01)	Функция	Минимум функции	X_1	X_2	Время выполнения метода (нс)
0,9		-1,499931	-1,510013	0,502494	9593706773
0,5		-1,491106	-1,384501	0,461048	2612978148
0,1		-1,480922	-1,668719	0,550458	1015548110

EPS	Минимум функции	X_1	X_2
0,001	-1,499457	-1,734954	-0,183184
0,01	-1,498744	-1,484763	0,489115

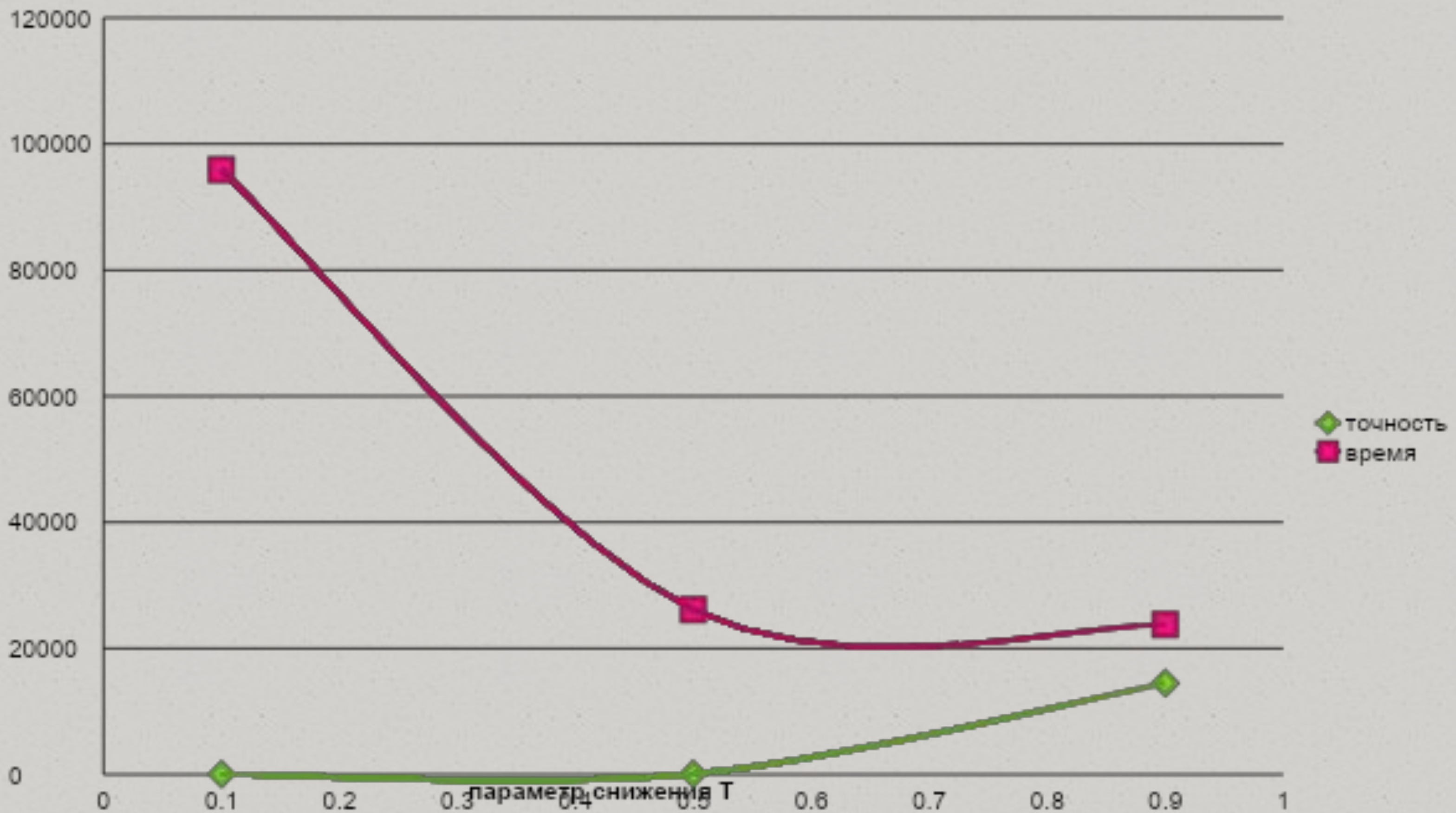
Сравнительный анализ скорости и точности метода имитации обжига в зависимости от максимального значения температуры



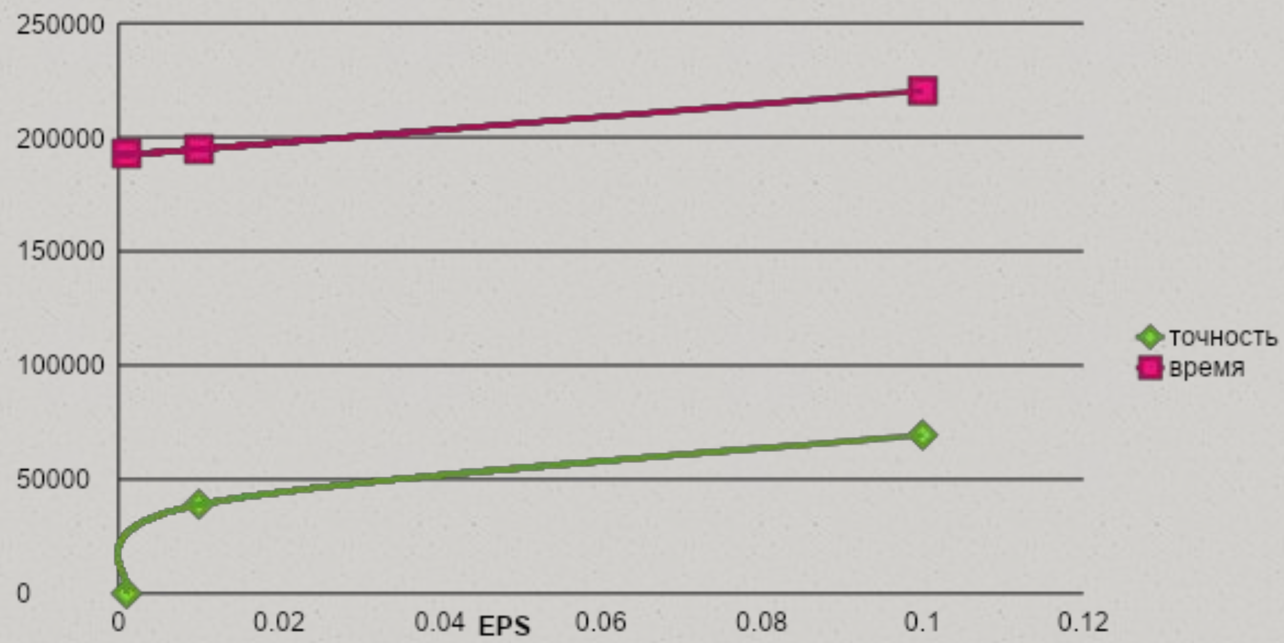
Сравнительный анализ скорости и точности метода имитации обжига в зависимости от количества циклов для каждой температуры



Сравнительный анализ скорости и точности метода имитации обжига в зависимости от параметра снижения температуры



Сравнительный анализ скорости и точности метода имитации обжига в зависимости от EPS



Вывод: оптимальными параметрами для метода имитации обжига являются:

- t_{max}= 40000;
- количество циклов = 100;
- параметр снижения температуры = 0.9;
- ε= 0.01.

Генетические алгоритмы

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

Основные понятия ГА:

- **Фитнесс-функция:** f(x).
- **Особь (**хромосома, индивид**):** $x = (x_1 x_2 x_3 \ldots x_n)$,
- **Ген** - бит строки x_i.
- **Популяция** - $X = \{x_i, i=1,\ldots,k\}$.
- Работа ГА - **смена поколений**.

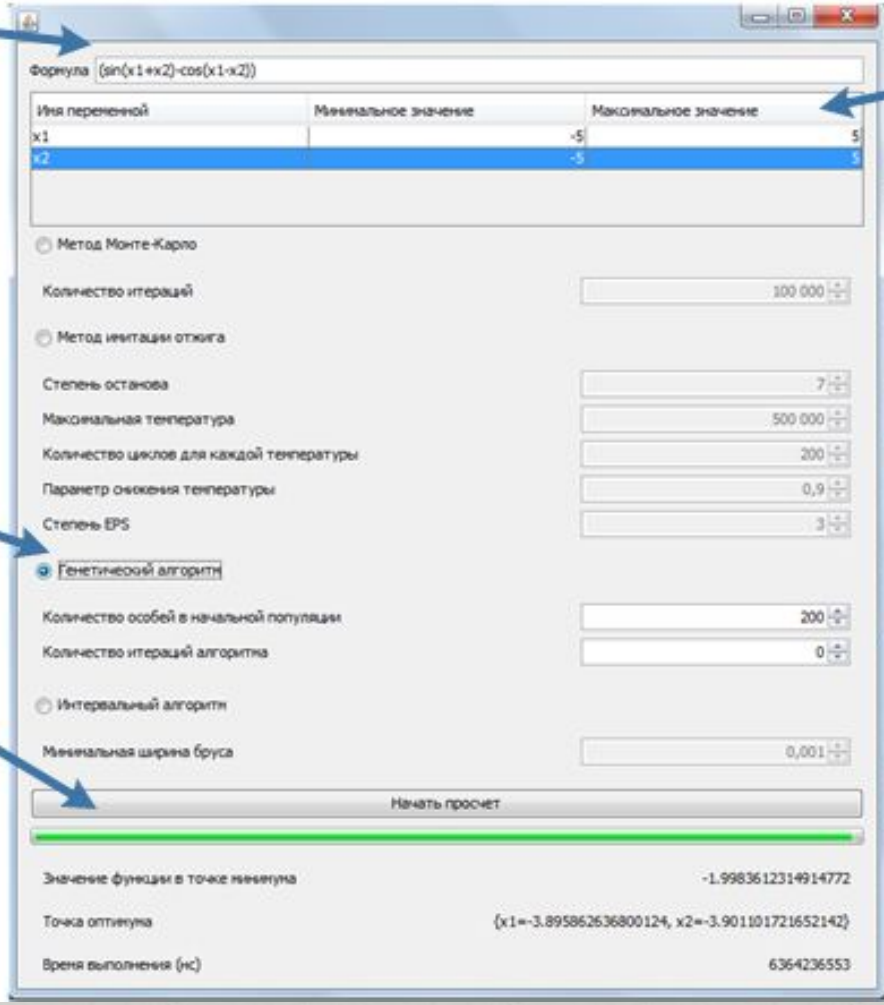
Алгоритм 3.

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 целевую функцию

Шаг 2: Задаём минимальное и максимальное значения переменных



Шаг 3: Выбираем «Генетический алгоритм»

Шаг 4: Задаём количество особей в начальной популяции и количество итераций алгоритма

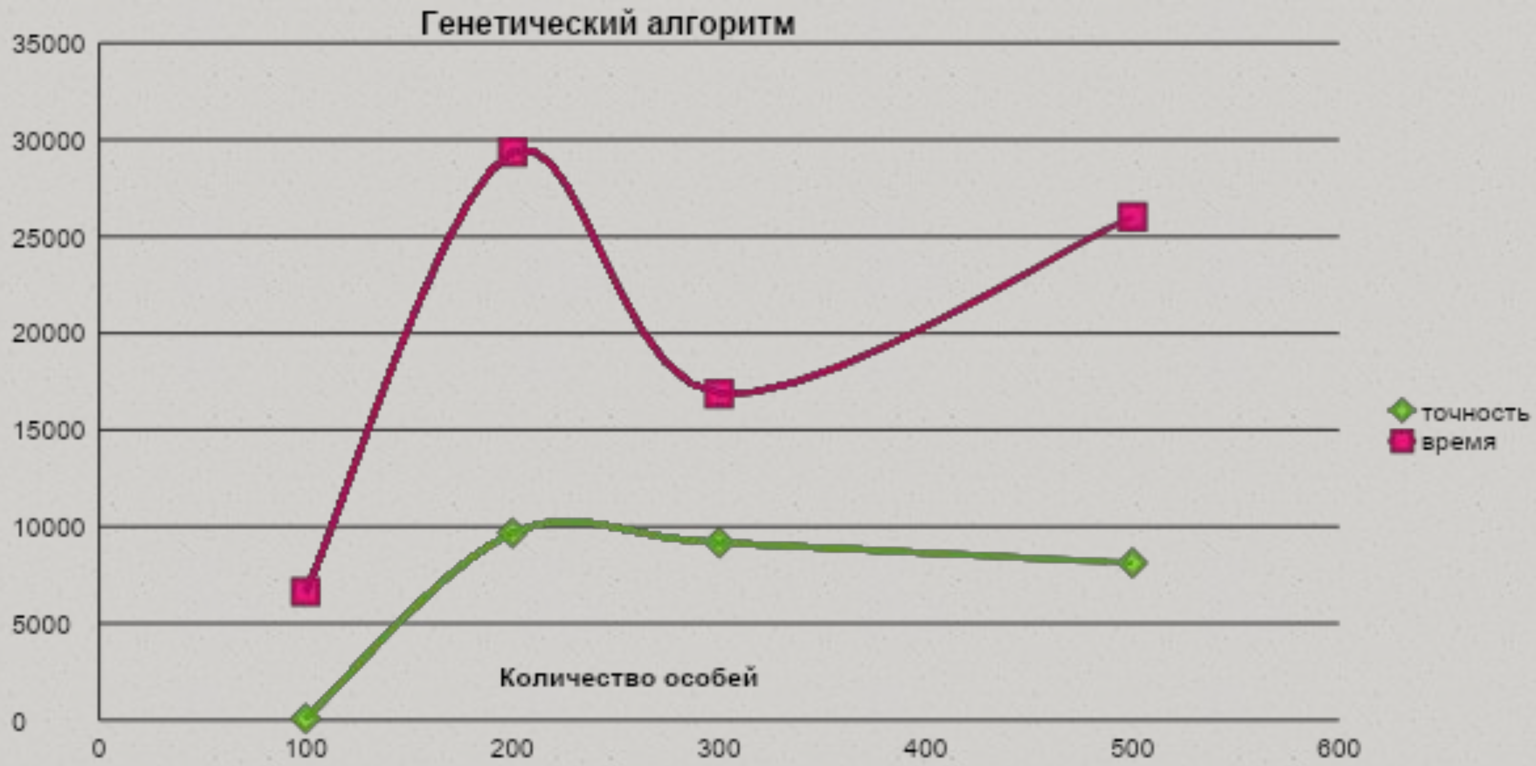
Шаг 5: Запускаем процесс вычисления

Сравнительный анализ скорости и точности ГА

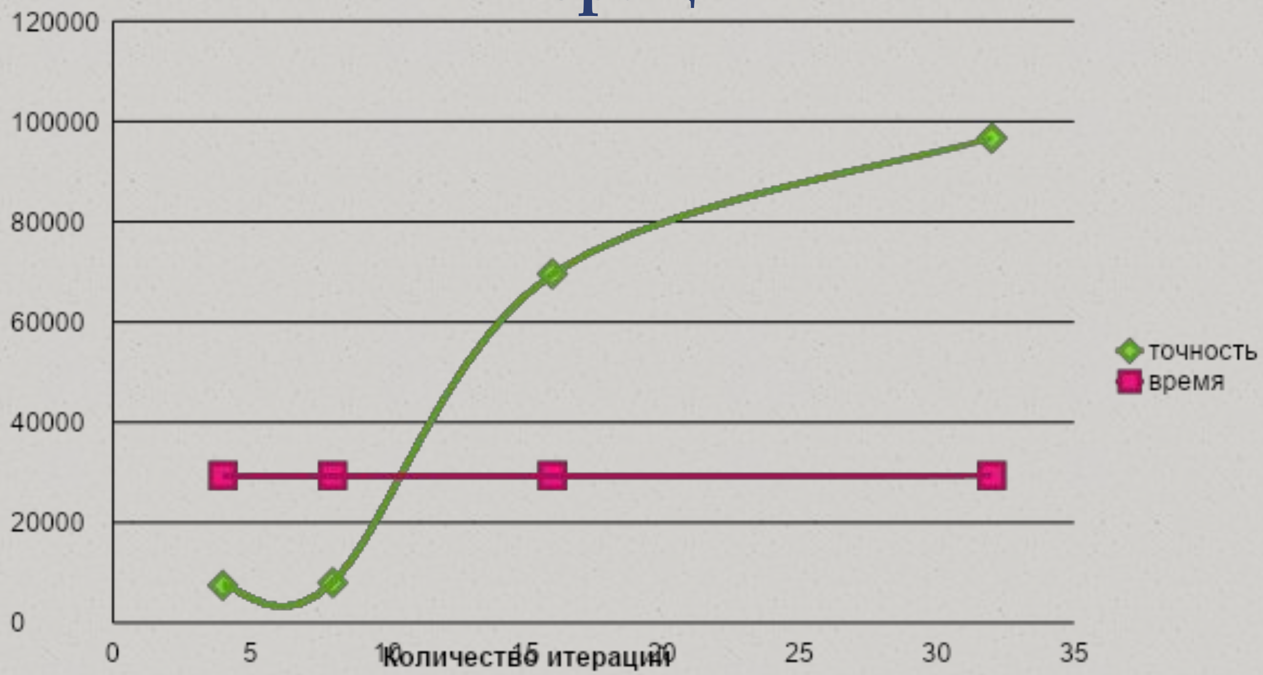
Целевая функция	Количество особей в начальной популяции	Количество итераций алгоритма	Значение функции в точке оптимума	Значения переменных	Время работы программы,нс
	100	2	-1.487776859	x_1= -1.4715869584 x_2= 0.4815268547	66341070
		4	-1.4877895415	x_1= -1.4726584913 x_2= 0.4835968214	66342154
		8	-1.4878659284	x_1= -1.4727849634 x_2= 0.4836958432	66343685
		16	-1.487874965	x_1= -1.4715869584 x_2= 0.4815268547	66345963
	200	2	-1.498468521	x_1= -1.4865247193 x_2= 0.4819658241	29360361
		4	-1.498657841	x_1= -1.4715868859 x_2= 0.4815268547	29362658
		8	-1.498752418	x_1= -1.4715869584 x_2= 0.4815268547	29364985
		16	-1.499856324	x_1= -1.4715869584 x_2= 0.4815635284	29367823

С ростом числа особей в начальной популяции и с ростом числа итераций алгоритма точность вычислений растёт, что, конечно, является положительным результатом, однако при этом растёт и время выполнения программы.

Сравнительный анализ скорости и точности ГА в зависимости от количества особей в начальной популяции



Сравнительный анализ скорости и точности ГА в зависимости от количества итераций



Вывод: получаем следующие оптимальные значения параметров алгоритма:

- количество особей в начальной популяции = 300
- количество итераций алгоритма = 16

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:

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

- Если

$$X_1, Y_1 \subset R^n$$

то

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

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

- 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: $[p] := [x]$, $c := \mathrm{mid}([p])$
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: $l := \arg\max \mathrm{wid}([p_i])$

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. $\underline{\tilde{f} < [f]_c}$, 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. $L := L + (p_i, [\underline{f}]([p_i]))$ – сохранение в списке

4.4. ВыполнятьБисекцию := Ложь

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

4.5.1. $(p, \underline{f})$:= 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

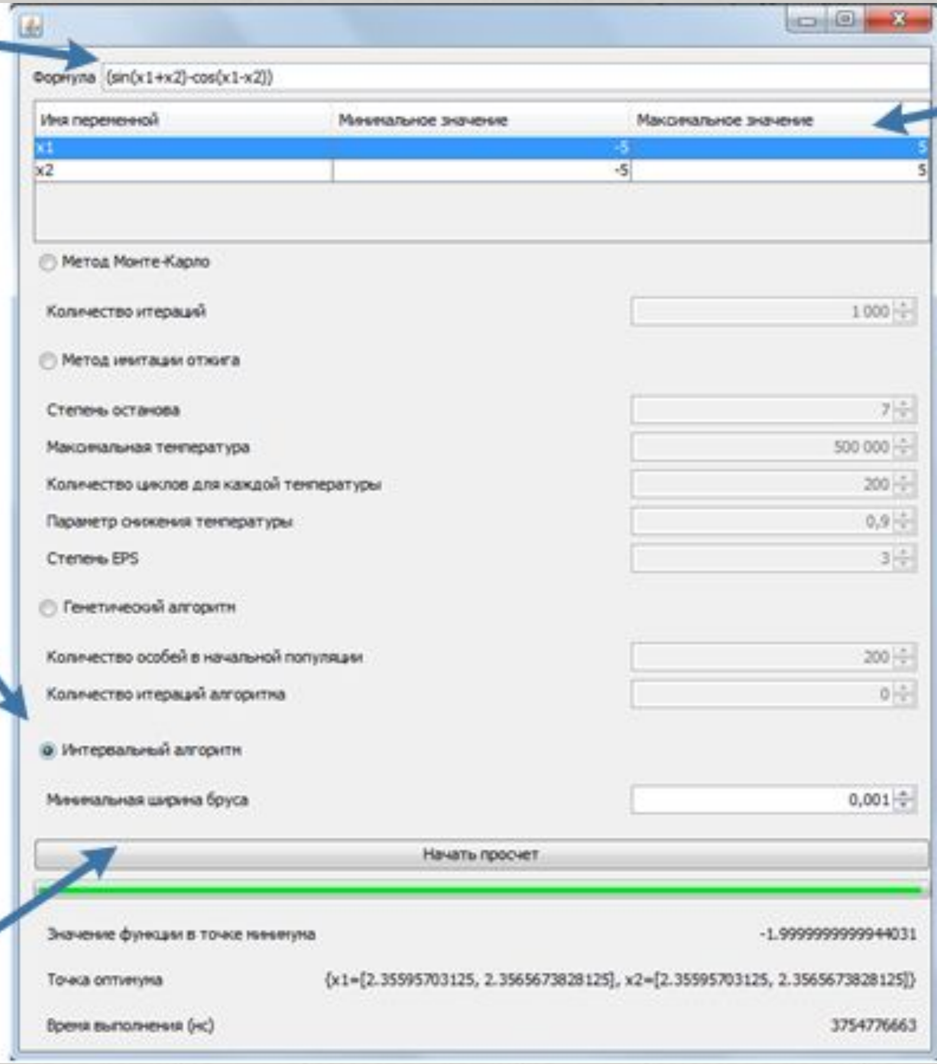
ПОКА (ВыполнятьБисекцию)

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

Программная реализация интервальных методов оптимизации

Шаг 1: Вводим целевую функцию

Шаг 2: Задаём минимальное и максимальное значения переменных



Шаг 3: Выбираем «Интервальный метод»

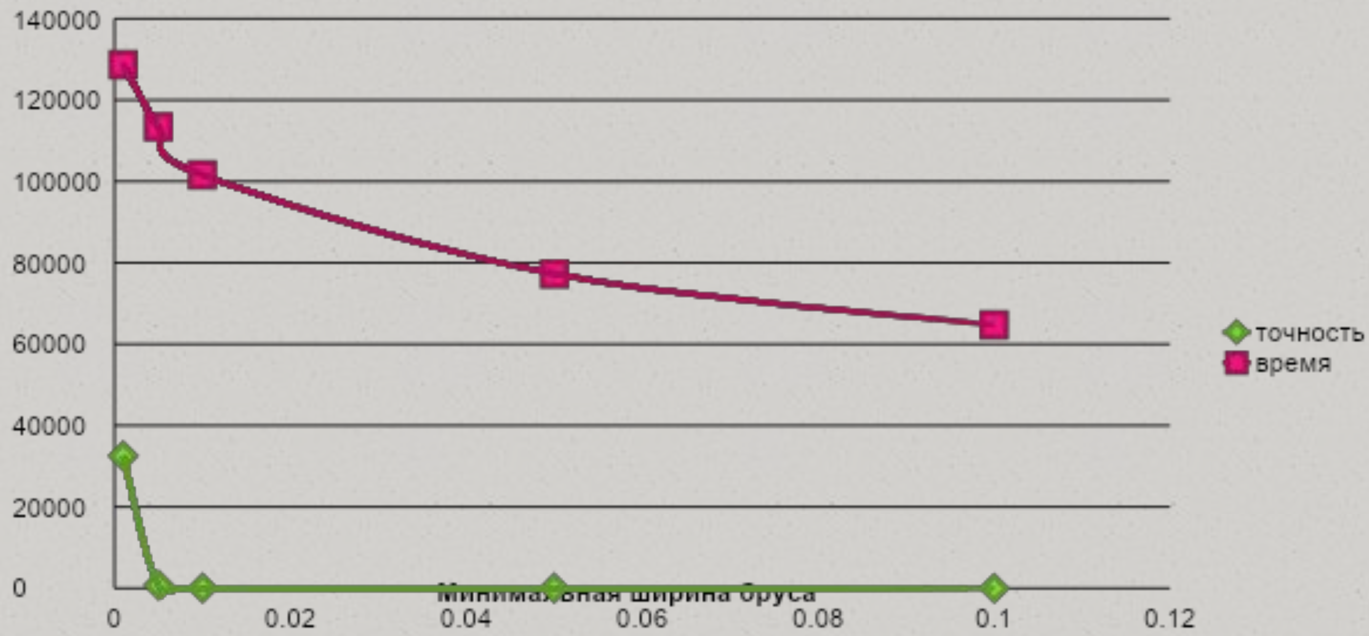
Шаг 4: Задаём минимальная ширма бруса

Шаг 5: Запускаем процесс вычисления

Сравнительный анализ скорости и точности ИА

Минимальная ширина бруса	Минимум функции и значения переменных
0,05	Минимум функции = -1,4878408 x1= -1,5227958 x2= 0,4639619 Время выполнения метода (нс):328396943
0,04	Минимум функции = -1,4989183 x1= -1,5023528 x2= 0,4968789 Время выполнения метода (нс):1276326871
0,03	Минимум функции = -1,4995258 x1= -1,4790195 x2= 0,5007679 Время выполнения метода (нс):2377426771
0,02	Минимум функции = -1,4997932 x1= -1,4973097 x2= 0,499912 Время выполнения метода (нс):12680595740
0,01	Минимум функции = -1,4998791 x1= -1,499992 x2= 0,5005327 Время выполнения метода (нс):211499245441

Сравнительный анализ скорости и точности ИА в зависимости от минимальной ширины бруса



Вывод: с увеличением минимальной ширины бруса уменьшается время работы, однако и точность тоже снижается. Оптимальная минимальная ширина бруса = 0,02.

Сравнительная таблица эффективности алгоритмов оптимизации

Целевая функция	Метод обжига	Монте-Карло	Генетические алгоритмы	Интервальный алгоритм
	Минимум функции= -1,495090966 x1= -1,52894452 x2= 0,547729271	Минимум функции= -1,4998664 x1= -1,5070133527 x2= 0,5088338996	Минимум функции= -1,4993594 x1= -1,546951234 x2= 0,563254891	Минимум функции= -1,4993594 x1= -1,546951234 x2= 0,563254891
	Минимум функции= -4,9989320881 x1= 0,8033830896 x2= 8,1581025431	Минимум функции= -4,99801404 x1= 0,7573183183 x2= 8,12728494035	Минимум функции= -4,9988421643 x1= 0,8002695431 x2= 8,1415692432	Минимум функции= -4,9988456854 x1= 0,8002684597 x2= 8,1415365821
	Минимум функции= -2,996650207 x1= 0,0050497306 x2= 0,03034247006	Минимум функции= -2,9997949001 x1= -0,002296043 x2= -0,0071462444	Минимум функции= -2,99895624879 x1= -0,0026325477 x2= -0,0156481398	Минимум функции= -2,9989598547 x1= -0,0026985471 x2= -0,0156332144

Сравнительная таблица эффективности алгоритмов оптимизации

- Для сравнения эффективности и скорости работы алгоритмов нами был произведен сравнительный анализ реализаций генетических алгоритмов и стохастических алгоритмов, реализованных ранее, - это метод имитации обжига, а также генетических алгоритмов. Это позволило нам прийти к выводам о высокой точности алгоритмов, гарантированном получении оптимума. Однако, интервальные алгоритмы не всегда самые быстрые – что может быть объяснено особенностями вычислений с интервалами.
- В большинстве случаев алгоритмы интервального анализа более точны, нежели реализованные ранее алгоритмы, кроме того, использование интервальных алгоритмов в глобальной оптимизации гарантирует получение оптимума, тогда как стохастические и генетические алгоритмы находят оптимум не всегда.
- Время работы алгоритма по сравнению с другими – более быстрое, реализация алгоритма значительно быстрее стохастических и генетических алгоритмов оптимизации, временные задержки могут быть связаны с особенностью интервального исчисления и переопределения функций.

Оптимальные параметры для методов оптимизации

Метод Монте - Карло	Метод имитации обжига	Генетический алгоритм	Интервальный метод
- число итераций 100000		- количество особей в начальной популяции = 300 - количество итераций алгоритма = 16	минимальная ширина бруса = 0,02

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 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!