Глобальные градиентные методы оптимизации

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Факультет компьютерных наук
Прикладная математика и информатика
г. Москва

Судаков Илья Александрович
Февраль 2023

1 Введение

Многие методы оптимизации (координатный спуск, градиентный спуск, метод сопряженного гадиента и др.) используют так называемый локальный поиск, т.е. поиск минимума одномерной функции вдоль луча в n-мерном евклидовом пространстве. При этом всегда находится локальный минимум. В проекте предлагается исследовать потенциал применения методов глобальной одномерной оптимизации для организации линейного поиска. В каких случаях удастся найти глобальной минимум многомерной функции, используя глобальный линейный поиск в контексте классических градиентных методов.

1.1 Задачи проекта

- Программная реализация методов глобального линейного поиска.
- Реализация алгоритма оптимизации многомерной функции.
- Экспериментальное и теоретическое исследование.

1.2 Ссылка на материалы

2 Координатный спуск

Координатный спуск — это алгоритм оптимизации, который последовательно проводит минимизацию функции вдоль координатных направлений. На очередной итерации, алгоритм фиксирует все координаты, кроме той, вдоль которой будет осуществляться поиск. Так как выбор направления на каждой итерации детерменирован, то нет необходимости наличия производных у функции.

Пусть целевая функция имеет вид: $F(\vec{x}) : \mathbb{X} \to \mathbb{R}$

Задача оптимизации задана в следующем виде: $F(\vec{x}) \to \min\limits_{\vec{x} \in \mathbb{X}}$

2.1 Алгоритм

- Задают начальное приближение и точность расчёта: X_0, ε
- Рассчитывают $x_i^{k+1} = \arg\min\limits_{y \in \mathbb{R}} f(x_1^{k+1}, \ldots, x_{i-1}^{k+1}, y, x_{i+1}^{k}, \ldots, x_n^{k})$
- Проверяют условие остановки (на усмотрение):
  - $|x^{[j]} - x^{[j+1]}| < \varepsilon$
  - $|F(x^{[j]}) - F(x^{[j+1]})| < \varepsilon$
  - иначе переходят к пункту 2

Таким образом, на каждой итерации будет выполнено $F(x^0) \geqslant F(x^1) \geqslant F(x^2) \geqslant \ldots$.

3 Градиентный спуск

Пусть целевая функция имеет вид: $F(\vec{x}) : \mathbb{X} \to \mathbb{R}$

Задача оптимизации задана в следующем виде: $F(\vec{x}) \to \min\limits_{\vec{x} \in \mathbb{X}}$

Основная идея метода заключается в том, чтобы идти в направлении наискорейшего спуска, а это направление задаётся антиградиентом $-\nabla F$

и $\vec{x}^{[j+1]} = \vec{x}^{[j]} - \lambda^{[j]} \nabla F\left(\vec{x}^{[j]}\right)$ где $\lambda^{[j]}$ задает скорость градиентного спуска и может быть выбрана как

- постоянная, тогда метод может не сходиться
- убывать по какому-то закону
- гарантировать наискорейший спуск

3.1 Алгоритм градиентного спуска

- Задают начальное приближение и точность расчёта: X_0, ε
- Рассчитывают $\vec{x}^{[j+1]} = \vec{x}^{[j]} - \lambda^{[j]} \nabla F\left(\vec{x}^{[j]}\right)$
- Проверяют условие остановки (на усмотрение):
  - $|x^{[j]} - x^{[j+1]}| < \varepsilon$
  - $|F(x^{[j]}) - F(x^{[j+1]})| < \varepsilon$
  - $\nabla F(x^{[i]}) < \varepsilon$
  - иначе переходят к пункту 2

Этот алгоритм будет реализован на python

3.2 Метод наискорейшего спуска

В случае наискорейшего спуска $\lambda^{[j]}$ определяется как:

$$\lambda^{[j]} = \text{argmin}_\lambda \, F\left(\vec{x}^{[j+1]}\right) = \text{argmin}_\lambda \, F\left(\vec{x}^{[j]} - \lambda \nabla F\left(\vec{x}^{[j]}\right)\right)$$

чтобы вычислить $\lambda^{[j]}$ будем использовать метод золотого сечения и алгоритм Moore-Skelboe

4 Метод золотого сечения

4.1 Описание

Метод золотого сечения — это эффективная реализации троичного поиска, служащего для нахождения минимума/максимума унимодальной функции на отрезке. Лучше тем, что на каждой итерации вычисляется значение в одной, а не двух точках.

Пусть хотим найти минимум на отрезке $[a, b]$, тогда разобьем его на 3 части 2 точками s_1, s_2, так чтобы при отсекании одного из крайних подотрезков, одна из точек осталась границей подотрезков, а соотношение длин подотрезков оставалось прежним.
$|[a, s_1]| = |[s_2, b]| = l_1$
$|[s_1, s_2]| = l_2$
тогда: $\frac{l_2}{l_1} = \frac{l_1 - l_2}{l_2} \Rightarrow l_1^2 - l_1 \cdot l_2 - l_2^2 = 0 \Rightarrow \frac{l_1}{l_2} = \frac{1+\sqrt{5}}{2} = \phi$

4.2 Алгоритм тернарного поиска

написать ли

4.3 Оценка сложности

Так как на каждой итерации длина рассматриваемого отрезка умножается на $\frac{l_1+l_2}{l_1+l_2+l_2} = \frac{3+\sqrt{5}}{4+2\sqrt{5}} = \phi^{-1}$, то для достижения точности δ потребуется $\frac{\ln(\frac{|[a,b]|}{\delta})}{\ln(\phi)} \approx 2\ln(\frac{|[a,b]|}{\delta})$ итераций.

4.4 Применение

А что если функция не унимодальная, тогда к чему сойдется этот метод? К точке, которая является одним из локальных минимумов функции либо одному из концов, в котором производная неположительная.

5 Глобальная одномерная оптимизация

Алгоритм поиска глобального минимума функции вдоль одного направления будет основываться на интервальном анализе. Перечислим основные определения и теоремы, которые нам пригодятся для посторения алгоритма.

5.1 Определение

Интервалом $[a, b]$ называется следующее множество:

$$[a, b] := \{x \in \mathbb{R} | a \leq x \leq b\}$$

5.2 Арифметические свойства интервалов

- $\mathbf{x} + \mathbf{y} = \underline{\mathbf{x}} + \underline{\mathbf{y}}, \overline{\mathbf{x}} + \overline{\mathbf{y}}$
- $\mathbf{x} - \mathbf{y} = \underline{\mathbf{x}} - \overline{\mathbf{y}}, \overline{\mathbf{x}} - \underline{\mathbf{y}}$
- $\mathbf{x} \cdot \mathbf{y} = min(\underline{\mathbf{x}}\underline{\mathbf{y}}, \underline{\mathbf{x}}\overline{\mathbf{y}}, \overline{\mathbf{x}}\underline{\mathbf{y}}, \overline{\mathbf{x}}\overline{\mathbf{y}}), max(\underline{\mathbf{x}}\underline{\mathbf{y}}, \underline{\mathbf{x}}\overline{\mathbf{y}}, \overline{\mathbf{x}}\underline{\mathbf{y}}, \overline{\mathbf{x}}\overline{\mathbf{y}})$
- $\mathbf{x} \div \mathbf{y} = \mathbf{x} \cdot [1/\overline{\mathbf{y}}, 1/\underline{\mathbf{y}}]$, если $0 \notin y$

5.3 Основная теорема интервальной арифметики

Пусть $f(x_1, ..., x_n)$ - рациональная функция вещественных аргументов $x_1, ..., x_n$, и для нее определен результат $\mathbf{F}(\mathbf{X_1}, ..., \mathbf{X_n})$ подстановки вместо аргументов интервалов их изменения $(X_1, ..., X_n) \subset \mathbb{R}^n$ и для $(X_1, ..., X_n)$ операциии выполняются по правилам интервальной арифметики. Тогда

$$\{f\{x_1, ..., x_n\} | x_1 \in \mathbf{X_1}, ..., x_n \in \mathbf{X_n}\} \subset \mathbf{F}(\mathbf{X_1}, ..., \mathbf{X_n})$$

5.4 Утверждение

Пусть $f(x_1, ..., x_n)$ - рациональная функция вещественных аргументов $x_1, ..., x_n$, и $\mathbf{F}(\mathbf{X_1}, ..., \mathbf{X_n})$ соответствующая ей интервальная функция, тогда выполняется монотонность по включению. Пусть $\mathbf{X_1}, ..., \mathbf{X_n}$ и $\mathbf{Y_1}, ..., \mathbf{Y_n}$, такие что $\mathbf{X_1} \subset \mathbf{Y_1}, ..., \mathbf{X_n} \subset \mathbf{Y_n}$, тогда

$$f\{\mathbf{X_1}, ..., \mathbf{X_n}\} \subset f\{\mathbf{Y_1}, ..., \mathbf{Y_n}\}$$

5.5 Определение

Пусть $N > 0$ натуральное число. Тогда если $\langle S_0, ..., S_{n-1} \rangle$ семейство непустых подмножеств множества S, тогда будем называть его разбиением множества S. В частности, обьединение $S_0, ..., S_{n-1}$ равно S , тогда последовательность $\langle S_0, ..., S_{n-1} \rangle$ является покрытием S .

5.6 Теорема

Рассмотрим задачу безусовной глобальной оптимизации. Пусть $\langle B_0, ..., B_{n-1} \rangle$ семейство множеств, содержащее глобальный минимум, такое что упорядочено по возрастанию нижней границы $f(B_i)$ для $i = 0, 1, 2, ..., N - 1$. Пусть U наименьшая из верхних значений функции для подмножеств $\langle f(B_0), ..., f(B_{n-1}) \rangle$. Тогда интервал $[lb(f(B_0)), U]$ содержит глобальным минимум μ.

Используя данные факты напишем алгоритм поиска глобального минимума вдоль направления.

6 Алгоритм Moore-Skelboe

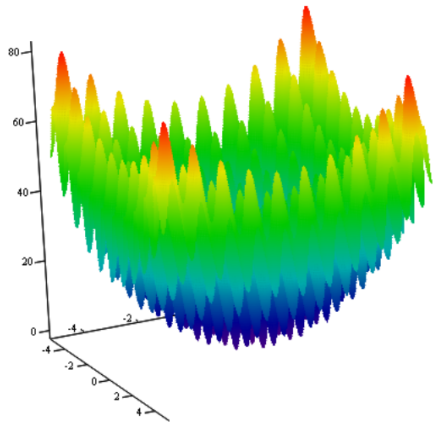
Ниже приведена реализация глобального линейного поиска, основанная на интервальном анализе

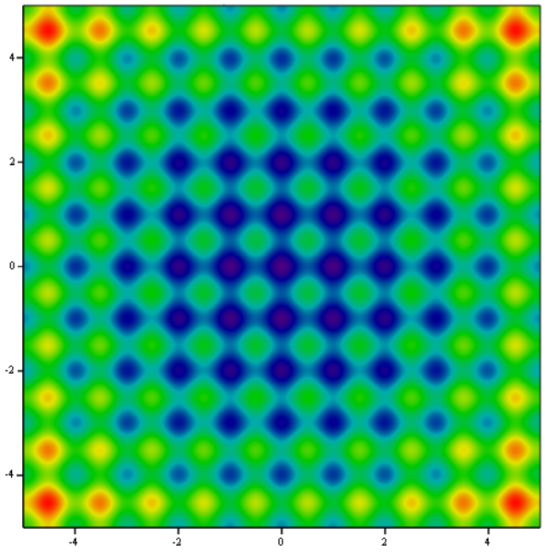
```
def MooreSkelboe(func_index, index, a, b, p, e_d, e_f):  # func_index(number of function in list of functions), index(index of direction),
    # a(left border), b(right border), p(current point), e_d(error of d), e_f(error of f)
    F = Functions[func_index * 2 + 1]
    interval_d = []
    for i in range(len(p)):
        interval_d.append(interval[p[i], p[i]])
    interval_d[index] = interval[a, b]

    interval_f = F(interval_d)
    set_of_intervals = [[interval_d, interval_f]]
    U = right(interval_f)
    w_f = right(interval_f) - left(interval_f)
    w_d = right(interval_d[index]) - left(interval_d[index])
    best_interval = set_of_intervals[0]
    while (w_d > e_d) | (w_f > e_f):
        set_of_intervals.pop(0)
        mid_p = mid(best_interval[0][index])
        interval_1 = best_interval[0].copy()
        interval_2 = best_interval[0].copy()
        interval_1[index] = interval[left(best_interval[0][index]), mid_p]
        interval_1_f = F(interval_1)
        interval_2[index] = interval[mid_p, right(best_interval[0][index])]
        interval_2_f = F(interval_2)
        U = min(U, right(interval_1_f))
        U = min(U, right(interval_2_f))

        for i in range(len(set_of_intervals)):
            if U < left(set_of_intervals[i][1]):
                set_of_intervals = set_of_intervals[:i]
                break

        set_of_intervals.append([interval_1, interval_1_f])
        set_of_intervals.append([interval_2, interval_2_f])

        set_of_intervals.sort(key=lambda item: left(item[1]))
        best_interval = set_of_intervals[0]
        w_f = right(best_interval[1]) - left(best_interval[1])
        w_d = right(best_interval[0][index]) - left(best_interval[0][index])

    best_point = []
    for i in range(len(p)):
        best_point.append(mid(best_interval[0][i]))

    return best_point

```

7 Тестирование

7.1 Функция Растригина

7.1.1 График функции при $n = 2$





7.1.2 Формула

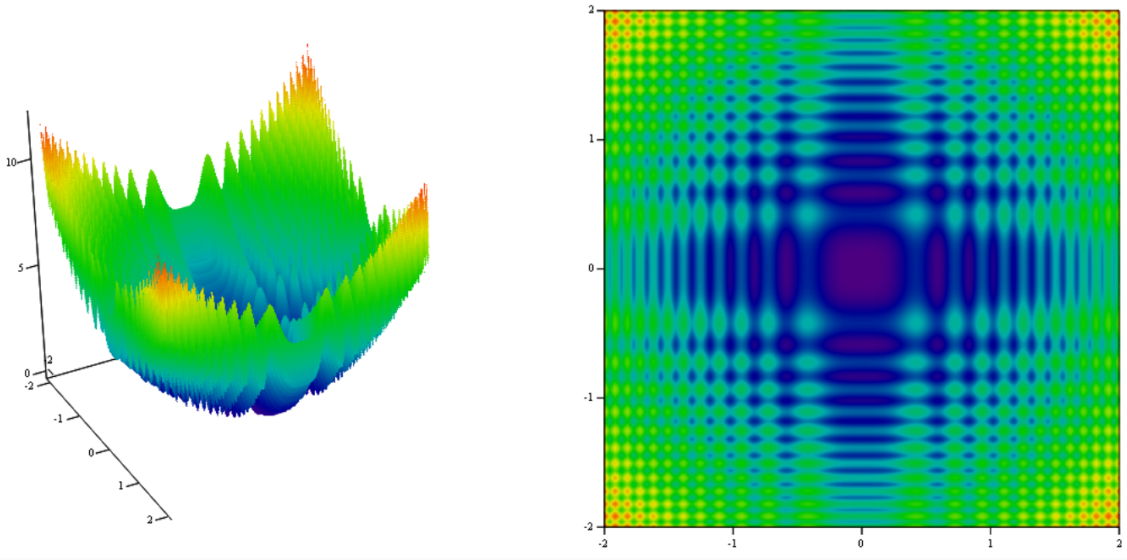
$$f(x) = 10n + \sum_{i=1}^{n} \left(x_i^2 - 10 \cdot \cos\left(2\pi \cdot x_i\right)\right)$$

7.1.3 Результаты тестирования

n	точность	метод	время	минимум
5	10^-3	Golden ratio	2.22 ms	4.974821275706137
	10^-3	Moore-Skelboe	178 ms	9.238348694395881e-05
10	10^-3	Golden ratio	4.78 ms	9.94964255141227
	10^-3	Moore-Skelboe	736 ms	0.00018476697390212848
20	10^-3	Golden ratio	13.3 ms	19.899285102824635
	10^-3	Moore-Skelboe	2.52 s	0.0003695339478184678
30	10^-3	Golden ratio	24.1 ms	29.84892765423696
	10^-3	Moore-Skelboe	5.38 s	0.0005543009217348072
40	10^-3	Golden ratio	43.2 ms	39.79857020564907
	10^-3	Moore-Skelboe	9.58 s	0.0007390678956511465
50	10^-3	Golden ratio	58.8 ms	49.748212757061154
	10^-3	Moore-Skelboe	14.9 s	0.0009238348695674858

7.2 Функция Растригина новгородская

7.2.1 График функции при $n = 2$



7.2.2 Формула

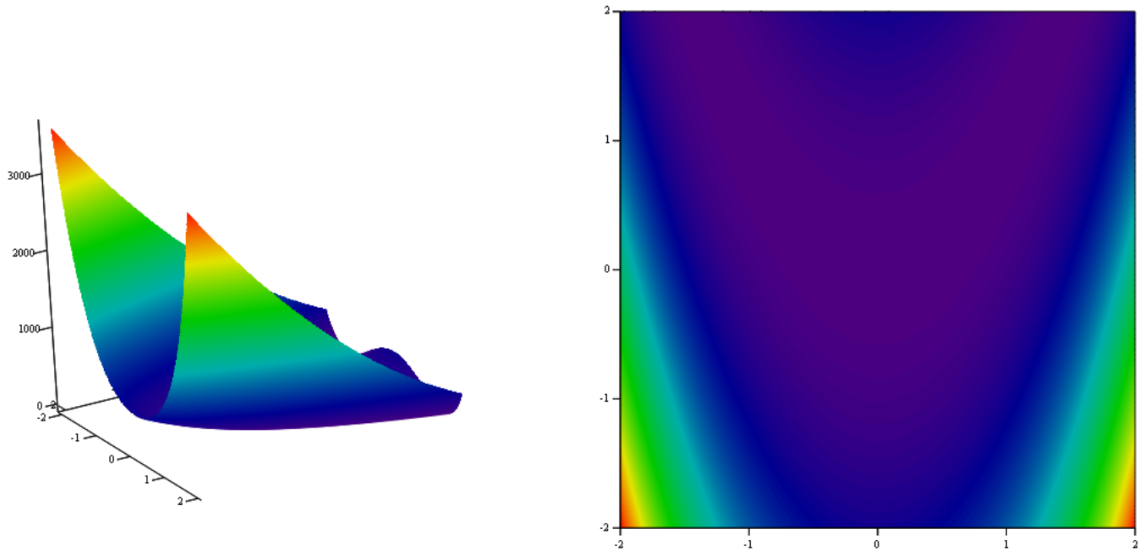
$$f(x) = n + \sum_{i=1}^{n} \left(x_i^2 - \cos\left(18 \cdot x_i^2\right)\right)$$

7.2.3 Результаты тестирования

n	точность	метод	время	значение
5	10^-3	Golden ratio	1.36 ms	1.542458003137213
	10^-3	Moore-Skelboe	151 ms	0.0007449817996270092
10	10^-3	Golden ratio	5.18 ms	3.0849160062744287
	10^-3	Moore-Skelboe	636 ms	0.0014899635992544624
20	10^-3	Golden ratio	20.8 ms	6.16983201254885
	10^-3	Moore-Skelboe	1.94 s	0.002979927198509369
30	10^-3	Golden ratio	37 ms	9.254748018823253
	10^-3	Moore-Skelboe	4.54 s	0.0044698907977642754
40	10^-3	Golden ratio	54.2 ms	12.339664025097658
	10^-3	Moore-Skelboe	7.79 s	0.005959854397019182
50	10^-3	Golden ratio	79.7 ms	15.424580031372061
	10^-3	Moore-Skelboe	11.6 s	0.0074498179962740885

7.3 Функция Розенброка

7.3.1 График функции при $n = 2$



7.3.2 Формула

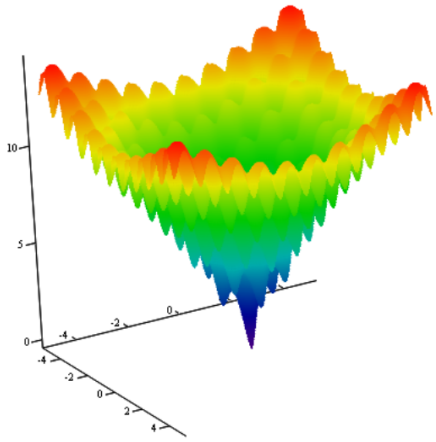
$$f(x) = \sum_{i=1}^{n-1} \left(100 \left(x_{i+1} - x_i^2 \right)^2 + (1 - x_i)^2 \right)$$

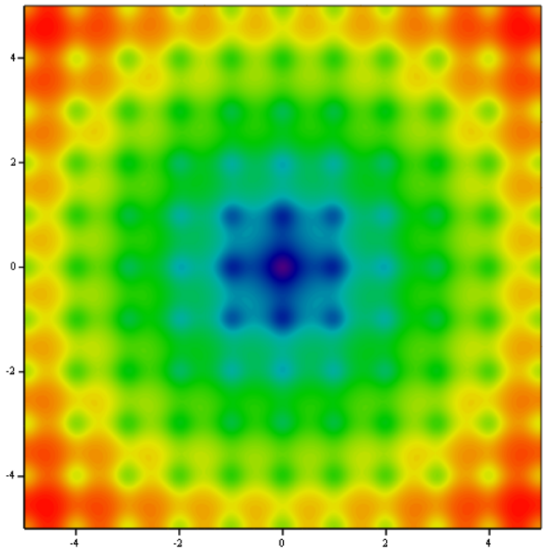
7.3.3 Результаты тестирования

n	точность	метод	время	значение
2	10^-3	Golden ratio	1.12 ms	0.16956582889643249
	10^-3	Moore-Skelboe	155 ms	0.16036252999621464
3	10^-3	Golden ratio	5.95 ms	0.259397310396006
	10^-3	Moore-Skelboe	3.99 s	0.11254106932659813
4	10^-3	Golden ratio	17 ms	0.28064340055794645
	10^-3	Moore-Skelboe	8.75 s	0.11734786681147619
5	10^-3	Golden ratio	27.3 ms	0.2830939134564683
	10^-3	Moore-Skelboe	15.6 s	0.11851270446807871
6	10^-3	Golden ratio	40.4 ms	0.29047947563212634
	10^-3	Moore-Skelboe	24.5 s	0.11879389607067276
7	10^-3	Golden ratio	49.2 ms	0.29064024190256904
	10^-3	Moore-Skelboe	35.6 s	0.1136410373197746
8	10^-3	Golden ratio	18.5 ms	0.33244082803407593
	10^-3	Moore-Skelboe	41.7 s	0.14015453688727936

7.4 Функция Экли

7.4.1 График функции при $n = 2$





7.4.2 Формула

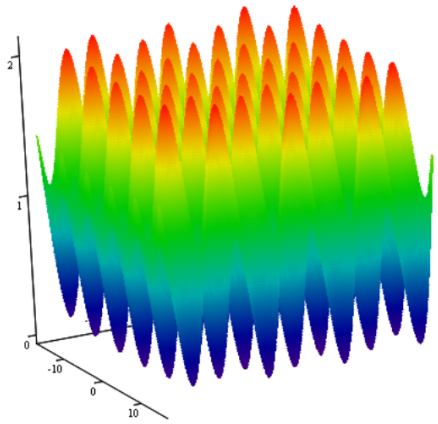
$$f(x) = 20 + e - 20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi \cdot x_i)}$$

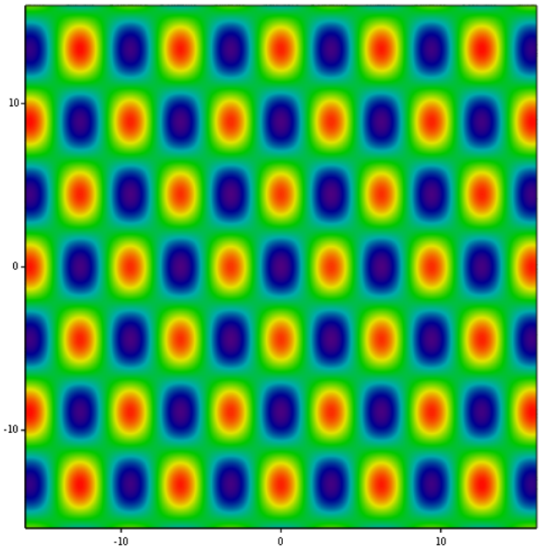
7.4.3 Результаты тестирования

n	точность	метод	время	значение
5	10^-3	Golden ratio	2.41 ms	-4.440892098500626e-16
	10^-3	Moore-Skelboe	553 ms	0.0012256630393632229
20	10^-3	Golden ratio	9.06 ms	3.5744545080266543
	10^-3	Moore-Skelboe	1.35 s	0.0012256630393632229
20	10^-3	Golden ratio	30.6 ms	3.5744545080266543
	10^-3	Moore-Skelboe	3.41 s	0.0012256630393632229
30	10^-3	Golden ratio	58.5 ms	3.574454508026655
	10^-3	Moore-Skelboe	6.23 s	0.0012256630393645551
40	10^-3	Golden ratio	84.1 ms	3.574454508026654
	10^-3	Moore-Skelboe	10.6 s	0.0012256630393632229
50	10^-3	Golden ratio	124 ms	3.5744545080266525
	10^-3	Moore-Skelboe	14.9 s	0.0012256630393618906

7.5 Функция Гриванка

7.5.1 График функции при $n = 2$





7.5.2 Формула

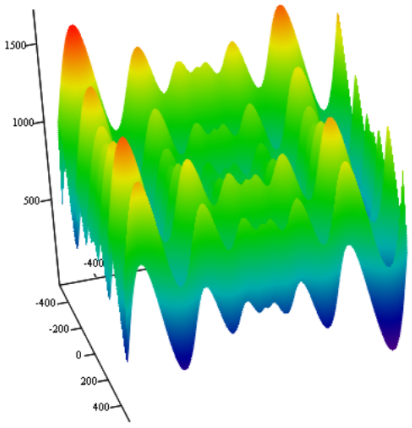
$$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

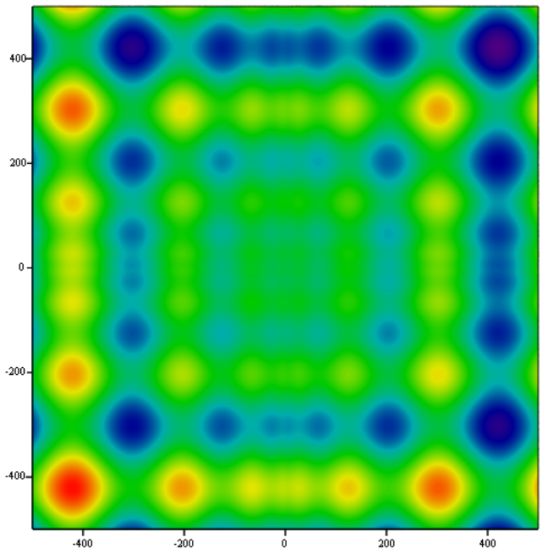
7.5.3 Результаты тестирования

n	точность	метод	время	значение
5	10^-3	Golden ratio	2.41 ms	0.009864698515380743
	10^-3	Moore-Skelboe	183 ms	1.0644240466817223e-07
10	10^-3	Golden ratio	6.05 ms	0.009864698515380743
	10^-3	Moore-Skelboe	739 ms	1.3662353537391425e-07
20	10^-3	Golden ratio	21.4 ms	0.009864698515380743
	10^-3	Moore-Skelboe	2.44 s	1.6799845647952338e-07
30	10^-3	Golden ratio	40.2 ms	0.009864698515380743
	10^-3	Moore-Skelboe	5.16 s	1.867295605917363e-07
40	10^-3	Golden ratio	59.3 ms	0.009864698515380743
	10^-3	Moore-Skelboe	9.09 s	2.0016648982768004e-07
50	10^-3	Golden ratio	84.4 ms	0.009864698515380743
	10^-3	Moore-Skelboe	14 s	2.1067470723501458e-07

7.6 Функция Швефеля

7.6.1 График функции при $n = 2$





7.6.2 Формула

$$f(x) = 418.9829n - \sum_{i=1}^{n} \left(x_i \sin\left(\sqrt{|x_i|} \right) \right)$$

7.6.3 Результаты тестирования

n	точность	метод	время	значение
2	10^-3	Golden ratio	721 µs	236.8766946858181
	10^-3	Moore-Skelboe	4.98 s	2.5455285594944144e-05
3	10^-3	Golden ratio	800 µs	355.3150420287272
	10^-3	Moore-Skelboe	9.41 s	3.8182928392416216e-05
4	10^-3	Golden ratio	1.44 ms	473.7533893716363
	10^-3	Moore-Skelboe	15 s	5.091057118988829e-05
5	10^-3	Golden ratio	1.4 ms	592.1917367145454
	10^-3	Moore-Skelboe	22 s	6.363821398736036e-05
6	10^-3	Golden ratio	2.19 ms	710.6300840574543
	10^-3	Moore-Skelboe	31.1 s	7.636585655745876e-05
7	10^-3	Golden ratio	1.88 ms	829.0684314003631
	10^-3	Moore-Skelboe	40 s	8.909349912755715e-05
8	10^-3	Golden ratio	4.92 ms	947.5067787432718
	10^-3	Moore-Skelboe	52 s	0.00010182114169765555

Список литературы

- Сергиенко А. Б. Тестовые функции для глобальной оптимизации.
- Пантелеев, Летова: Методы оптимизации в примерах и задачах.
- Ratschek, Rokne. New Computer Methods for Global Optimization.

8 Первые выводы