Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет
по лабораторной работе

Реализация алгоритмов многомерной оптимизации

по дисциплине «**Методы оптимизации**»

Авторы: Асмирко Антон, Герасимов Михаил, Пушкарев Глеб

Группа: M3235

Факультет: ФИТиП

Преподаватель: Корсун Мария Михайловна

Санкт-Петербург 2021

Условия выполнения лабораторной работы №2

1. Реализовать алгоритмы:

- метод градиентного спуска;
- метод наискорейшего спуска;
- метод сопряженных градиентов.

Оцените, как меняется скорость сходимости, если для поиска величины шага использовать различные методы одномерного поиска.

2. Проанализируйте траектории методов для нескольких квадратичных функций: придумайте две-три квадратичные двумерные функции, на которых работа каждого из методов будет отличаться. Нарисуйте графики с линиями уровня функций и траекториями методов (для отрисовки можно воспользоваться сторонними программами, для получения бонусных баллов программу отрисовки реализовать самостоятельно).

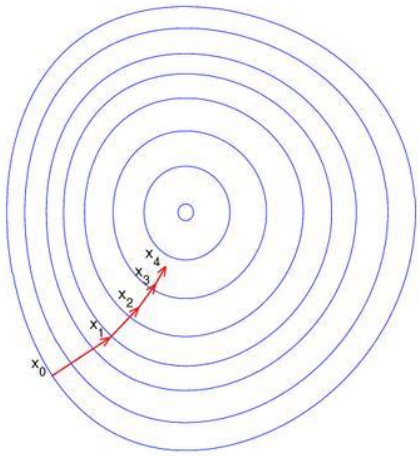


Рис. Пример изображения линий уровня и траектории метода

3. Исследуйте, как зависит число итераций, необходимое методам для сходимости, от следующих двух параметров:
а) числа обусловленности $k \geq 1$ оптимизируемой функции;
б) размерности пространства n оптимизируемых переменных.

Для этого для заданных параметров n и k сгенерируйте случайным образом квадратичную задачу размера n с числом обусловленности k и запустите на ней методы с некоторой заданной точностью. Замерьте число итераций $T\ (n, k)$, которое потребовалось сделать методу до сходимости.

4. По результатам выполнения лабораторной работы необходимо подготовить отчет. Отчет должен содержать все исследования, проведенные в п.п. 1-3, а также соответствующие выводы.
Для разработанного программного кода в отчете привести код основных модулей, диаграмму классов, сделать текстовое описание.

Требования к программному коду (вычислительные алгоритмы)

1. Рекомендуется использовать языки программирования: C++, C#, Java.
2. Рекомендуется придерживаться основных положений ООП при разработке.
3. Рекомендуется выполнять документирование программного кода.

Оценка результатов

Задание	*Результат (в виде коэффициента)*
Сдача в срок	0.15
Основные численные результаты, выводы, защита	0.0 – 0.55
Программная реализация и индивидуальный код	0.0 – 0.25
Грамотность изложения и общее качество отчета	0.0 – 0.05
Дополнительное задание (графика)	0.0 – 0.3

Срок сдачи второй лабораторной работы – до 16.04.2021 (включительно).

Вычислительные схемы алгоритмов

Рассматриваемые методы являются итерационными и задаются следующей схемой:

$$x^{k+1} = x^k + a_k p^k, \;\; k > 0$$
$$f(x^{k+1}) < f(x^k), \;\; k \geq 1$$

где направление убывания p^k определяется с учетом информации о частных производных функции $f(x)$, a_k – величина шага, причем $a_k > 0$

Так как функция предполагается дифференцируемой, то в качестве прекращения итераций можно выбрать условие $\|\nabla f(x^k)\| < \varepsilon$

1. Градиентный спуск — метод нахождения локального минимума или максимума функции с помощью движения вдоль градиента. Для минимизации функции в направлении градиента используются методы одномерной оптимизации.

   Основная идея метода градиентного спуска заключается в том, чтобы осуществлять оптимизацию функции в направлении наискорейшего спуска, которое задается антиградиентом.

$$-\nabla f = -\begin{pmatrix} \dfrac{\partial f(x)}{\partial x_1} \\ \dfrac{\partial f(x)}{\partial x_2} \\ \vdots \\ \dfrac{\partial f(x)}{\partial x_n} \end{pmatrix}$$

На каждом шаге метода предполагается, что $p^k = -\nabla f(x^k)$

Если $\nabla f(x^k) \neq 0$, то выполняется следующие условие: $(\nabla f(x^k), \; p^k) < 0$
Значит по теореме о достаточном условии направления убывания направление вектора p^k является направлением убывания, причем в малой окрестности точки x^k направление p^k обеспечивает наискорейшее убывание этой функции. Поэтому можно найти такое $a_k > 0$, что будет выполняться условие $f(x^{k+1}) < f(x^k), \;\; k \geq 1$.

Алгоритм метода градиентного спуска следующий:

Шаг 1. Задать параметр точности $\varepsilon > 0$, начальный шаг $a > 0$, выбрать $x \in E_n$ и вычислить $f(x)$. Перейти к шагу 2.
Шаг 2. Вычислить $\nabla f(x)$ и проверить условие достижения точности $\|\nabla f(x)\| < \varepsilon$. Если оно выполнено, то вычисления завершить, полагая $x^* = x, \;\; f^* = f(x)$. Иначе перейти к шагу 3.
Шаг 3. Найти $y = x - a\nabla f(x)$ и $f(y)$. Если $f(y) < f(x)$, то положить $x = y, \;\; f(x) = f(y)$ и перейти к шагу 2, иначе – к шагу 4.
Шаг 4. Положить $a = a/2$ и перейти к шагу 3.

Необходимо заметить, что вблизи стационарной точки функции $f(x)$ величина $\|\nabla f(x)\|$ становится малой. Это часто приводит к замедлению сходимости последовательности $\{x^k\}$. Поэтому в основной формуле $x^{k+1} = x^k + a_k p^k, \ \ k > 0$ иногда вместо антиградиента используют единичный вектор, который имеет схожее направление с антиградиентом.

$$p^k = \frac{-\nabla f(x^k)}{\|-\nabla f(x^k)\|}$$

2. Метод наискорейшего спуска – это модификация градиентного спуска. Согласно этому методу после вычисления в начальной точке градиента функции делают в направлении антиградиента не маленький шаг, а передвигаются до тех пор, пока функция убывает. Достигнув точки минимума на выбранном направлении, снова вычисляют градиент функции и повторяют описанную процедуру. При этом градиент вычисляется гораздо реже, поскольку это происходит только при смене направлений движения. В методе наискорейшего спуска аналогично методу градиентного спуска вектор $p^k = -\nabla f(x^k)$, но величина a_k находится в результате решения задачи одномерной оптимизации:

$$\Phi_k(a) \to min, \ \ \Phi_k(a) = f\left(x^k - a\nabla f(x^k)\right), \ \ a > 0$$

То есть на каждой итерации в направлении антиградиента -$\nabla f(x^k)$ совершается исчерпывающий спуск

Алгоритм метода наискорейшего спуска следующий:

Шаг 1. Задать параметр точности $\varepsilon > 0$, выбрать $x \in E_n$ и вычислить $f(x)$. Перейти к шагу 2.

Шаг 2. Вычислить $\nabla f(x)$ и проверить условие достижения точности $\|\nabla f(x)\| < \varepsilon$. Если оно выполнено, то вычисления завершить, полагая $x^* = x, \ \ f^* = f(x)$. Иначе перейти к шагу 3.

Шаг 3. Решить задачу одномерной оптимизации для $x^k = x$, то есть найти a^*. Положить $x = x - a^* \nabla f(x)$ и перейти к шагу 2.

Стоит заметить, что решение задачи одномерной оптимизации можно получить одним из пройденных для одномерной минимизации способов. Если, кроме того, функция $f(x)$ квадратична, то для величины шага исчерпывающего спуска можно применить формулу

$$a_k = -\frac{(\nabla f(x^k),\ p^k)}{(Ap^k,\ p^k)} = -\frac{(Ax^k + b,\ p^k)}{(Ap^k,\ p^k)} \quad \text{(по теореме)},$$

положив $p^k = -\nabla f(x^k)$

3. Метод сопряженных градиентов – метод нахождения локального экстремума функции на основе информации о её значениях и её градиенте. В случае квадратичной функции в $\mathbb{R}^n$ минимум находится не более чем за n шагов.

При использовании методов градиентного и наискорейшего спуска в итерационной процедуре в качестве направления убывания функции использовалось направление антиградиента, однако такой выбор направления убывания не всегда бывает удачным. В частности, для плохо обусловленных задач минимизации направление антиградиента в точке x^k может значительно отличаться от направления к точке минимума x^*. В результате траектория приближения к точке минимума имеет зигзагообразный характер.
Воспользуемся другим подходом, идея которого была изложена при построении метода сопряженных направлений. Будем определять направления спуска не только через вектор антиградиента, но и через направление спуска на предыдущем шаге. Это позволит более полно учесть особенности функции при нахождении ее минимума.

Используется следующий итерационный процесс:

$$x^{k+1} = x^k + a_k p^k, \;\; k = 0,1,2 \ldots; \;\; x^0 \in E_n, \;\; p^0 = -\nabla f(x^0)$$

$$p^{k+1} = -\nabla f(x^{k+1}) + \beta_k p^k, \;\; k = 0,1,2 \ldots,$$

$$\beta_k = \frac{(A\nabla f(x^{k+1}),\, p^k)}{(Ap^k,\, p^k)}$$

$$a_k = -\frac{(\nabla f(x^k),\, p^k)}{(Ap^k,\, p^k)} = -\frac{(Ax^k + b,\, p^k)}{(Ap^k,\, p^k)} \quad \text{(по теореме)}$$

Величина шага a_k находится из условия исчерпывающего спуска по направлению p^k. Далее, после вычисления очередной точки x^{k+1} новое направление поиска p^{k+1} находится по формуле, отличной от антиградиента, где коэффициенты β_k выбираются так, чтобы при минимизации квадратичной функции $f(x)$ с положительно определенной матрицей A получалась последовательность A–ортогональных векторов $p^0, p^1, p^2, \ldots$

Из условия $(Ap^{k+1},\, p^k) = 0$ мы имеем, что $\beta_k = \frac{(A\nabla f(x^{k+1}),\, p^k)}{(Ap^k,\, p^k)}$

Итерационный процесс можно упростить, а именно нахождение a_k и β_k:

$$\beta_k = \frac{\|\nabla f(x^{k+1})\|^2}{\|\nabla f(x^k)\|^2}$$

$$a_k = \frac{\|\nabla f(x^k)\|^2}{(Ap^k,\, p^k)}$$

$$f(x^k + a_k p^k) = \min_{a>0} f(x^k + a_k p^k)$$

Следует отметить, что выражение для коэффициента β_k не содержит в явном виде матрицу A квадратичноq формы. Поэтому метод сопряженных градиентов может применяться для минимизации неквадратичных функций.

Итерационная схема с упрощениями может не приводить к точке минимума неквадратичной функции $f(x)$ за конечное число итераций.

Так же важно помнить, что точное определение a_k из условия возможно лишь в редких случаях, а вектора p^kна образуют, вообще говоря, A −ортогональную систему относительно какой-либо матрицы A. Поэтому реализация каждой итерации метода будет сопровождаться неизбежными погрешностями. Эти погрешности, накапливаясь, могут привести к тому, что векторы p^k перестанут указывать направление убывания функции и сходимость метода может нарушаться. Поэтому в методе сопряженных градиентов применяется практический прием − через каждые Nшагов производят обновление метода, полагая $\beta_{m\cdot N} = 0, \ \ m = 1,2,3, \dots$. Номера mN называют моментами обновления метода, или *рестарта*. Часто полагают $N = n$ − размерности пространства E_n. Если $N = 1$, то получается частный случай метода сопряженных градиентов − метод наискорейшего спуска. Вблизи точки минимума дважды дифференцируемая функция с положительно определенной матрицей Гессе $H(x^*)$, как правило, достаточно хорошо аппроксимируется квадратичной функцией. Поэтому можно надеяться на хороший результат применения метода сопряженных градиентов для функций такого вида.

Алгоритм метода сопряженных градиентов следующий:

Шаг 1. Задать параметр точности $\varepsilon > 0$, выбрать $x \in E_n$, вычислить $f(x)$ и $\nabla f(x)$. Перейти к шагу 2.
Шаг 2. Вычислить p и перейти к шагу 3.
Шаг 3. Решить задачу одномерной минимизации $f(x + ap) \to min$. Перейти к шагу 4.
Шаг 4. Вычислить $\nabla f(x)$ и проверить условие достижения точности $\|\nabla f(x)\| < \varepsilon$. Если оно выполнено, то вычисления завершить, полагая $x^* = x, \ \ f^* = f(x)$. Иначе перейти к шагу 5.
Шаг 5. Проверить условия рестарта, по необходимости произвести обновление метода. Перейти к шагу 6.
Шаг 6. Вычислить β и новое направление спуска p. Перейти к шагу 3.

Исследование траекторий методов на квадратичных функциях

При исследовании критерием остановки принималось условие $\|\nabla f(x^k)\| < \varepsilon = 10^{-5}$. Начальный шаг $\alpha = 100$. Для каждого метода и функции приведены графики с линиями уровней функций и направлениями методов.

I. $f(x,y) = 35 * x * x + 71 * x * y + 45 * y * y - 8 * x + 62 * y + 11$

II. $f(x,y) = 15 * x * x + 3 * y * y - 7 * x + 11 * y - 8$

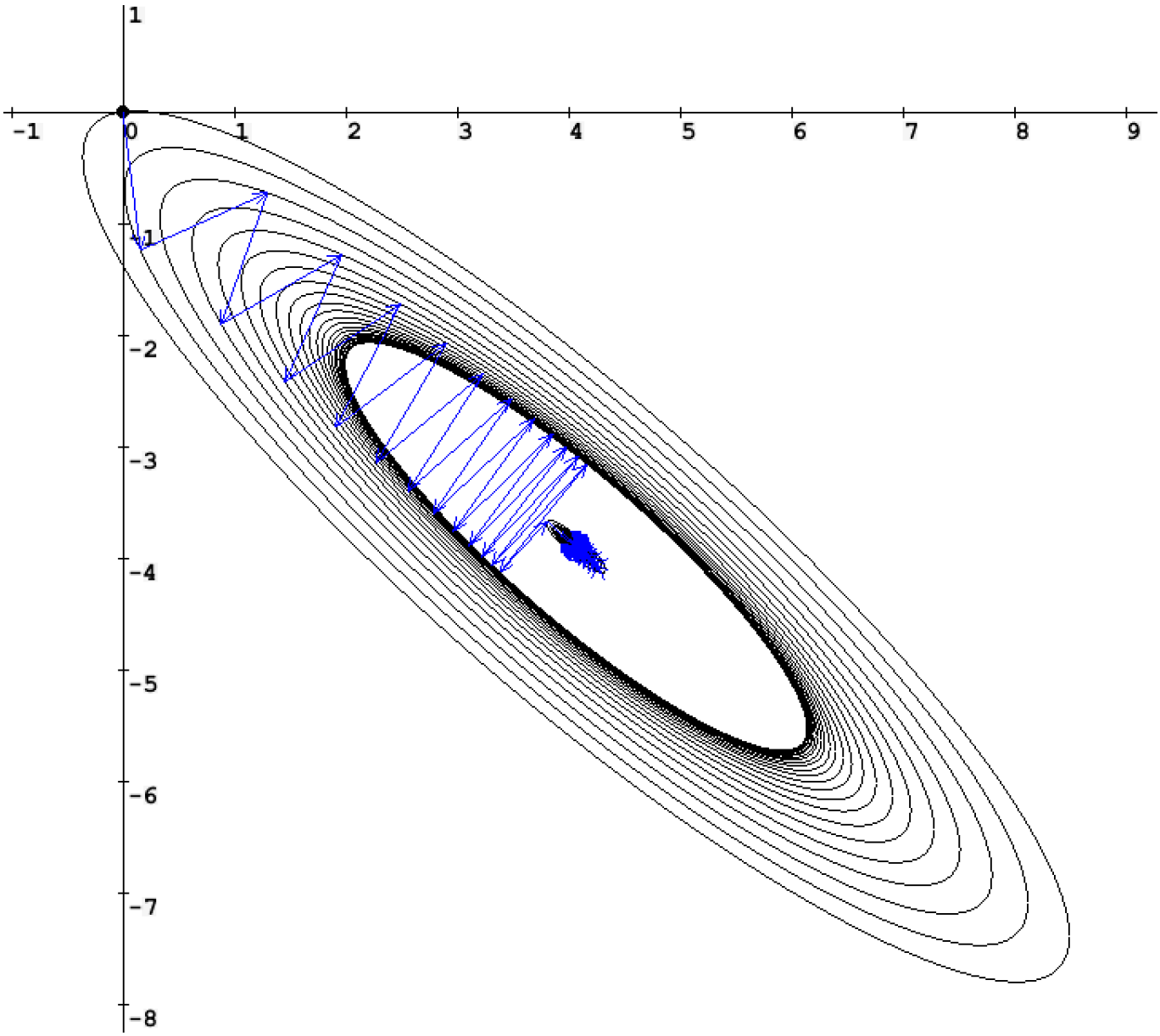


График 1: Function1, Gradient Descent

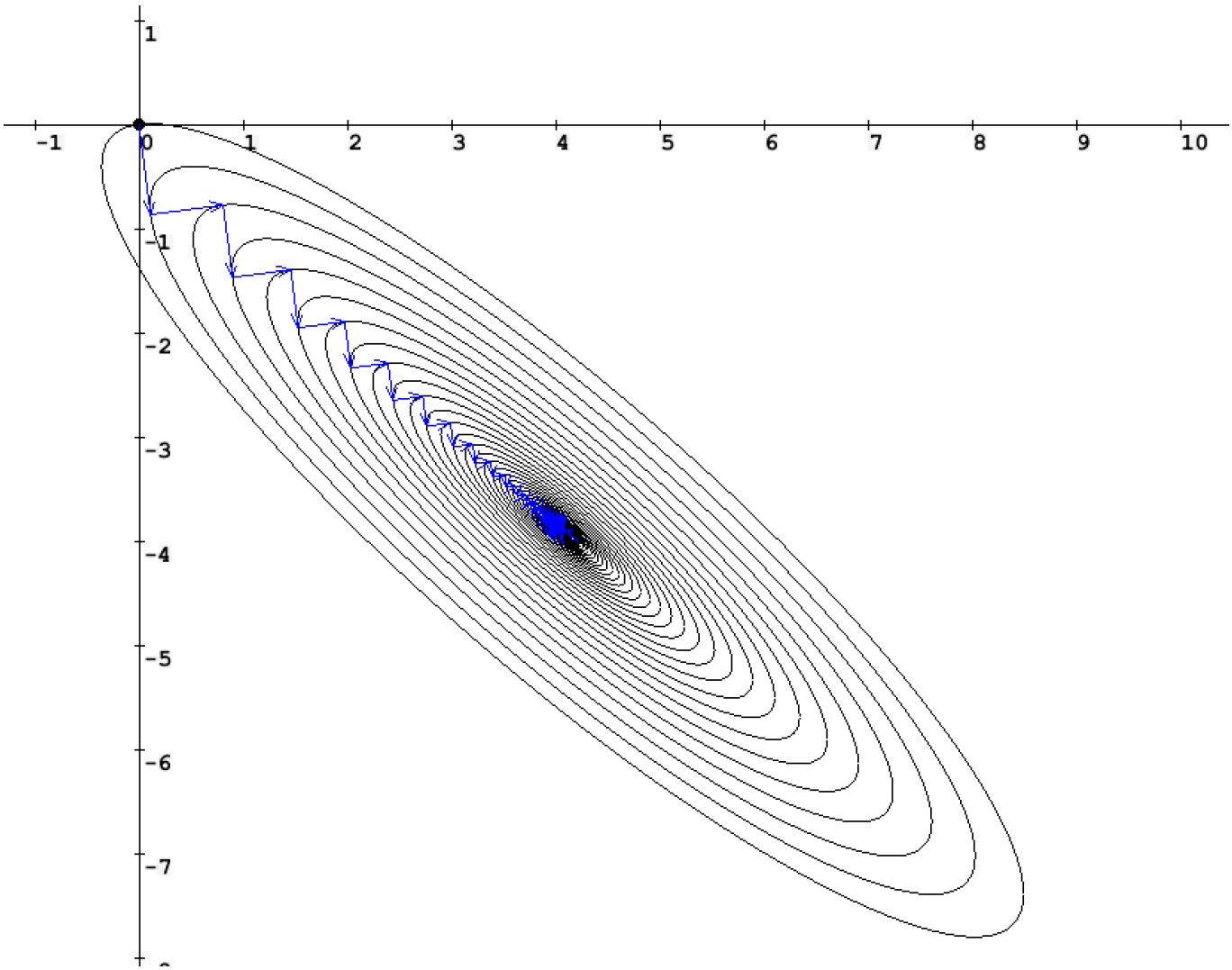


График 2 Function1, Fastest Gradient (Dichotomy)

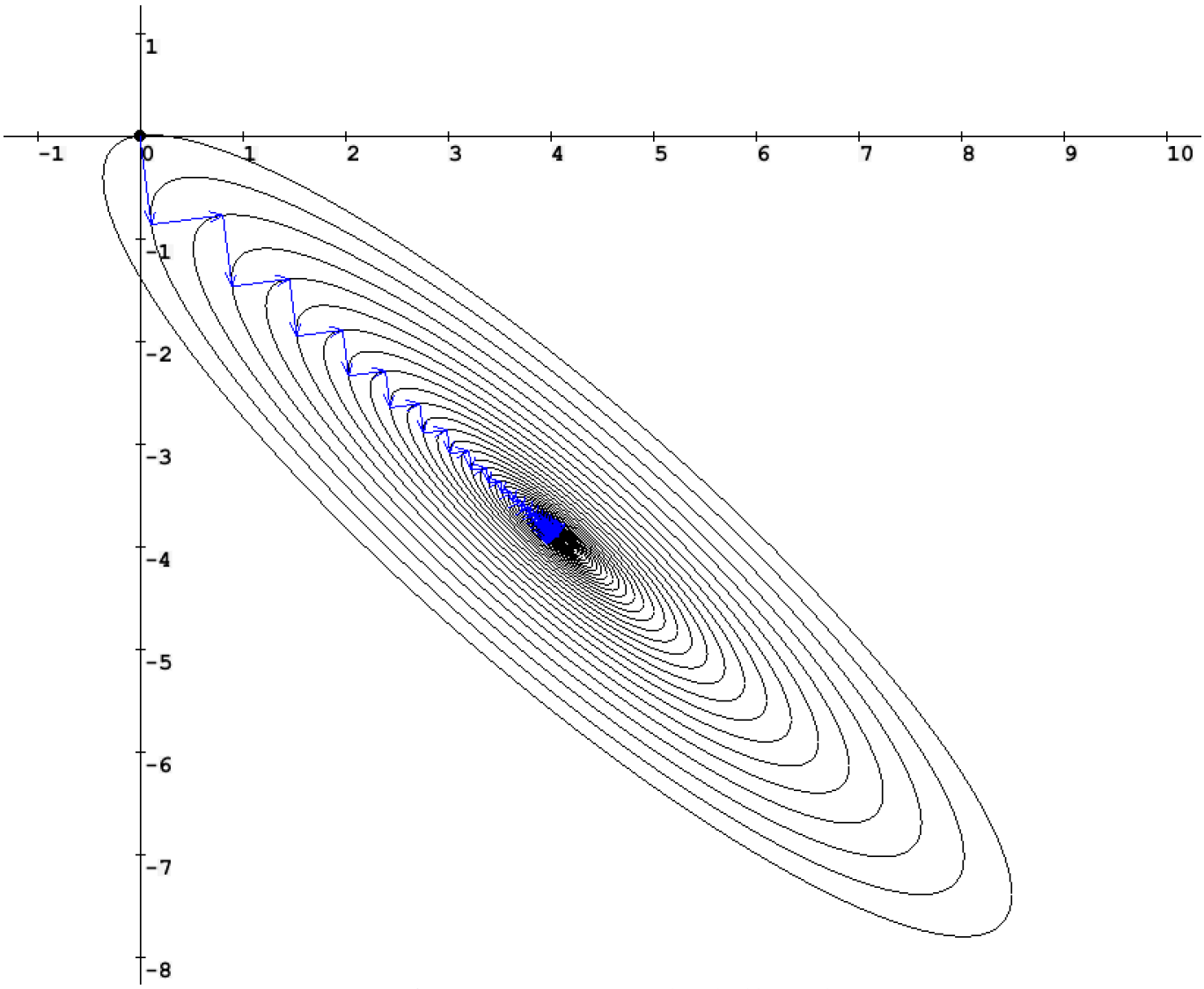


График 3 Function1, Fastest Gradient (Golden ratio)

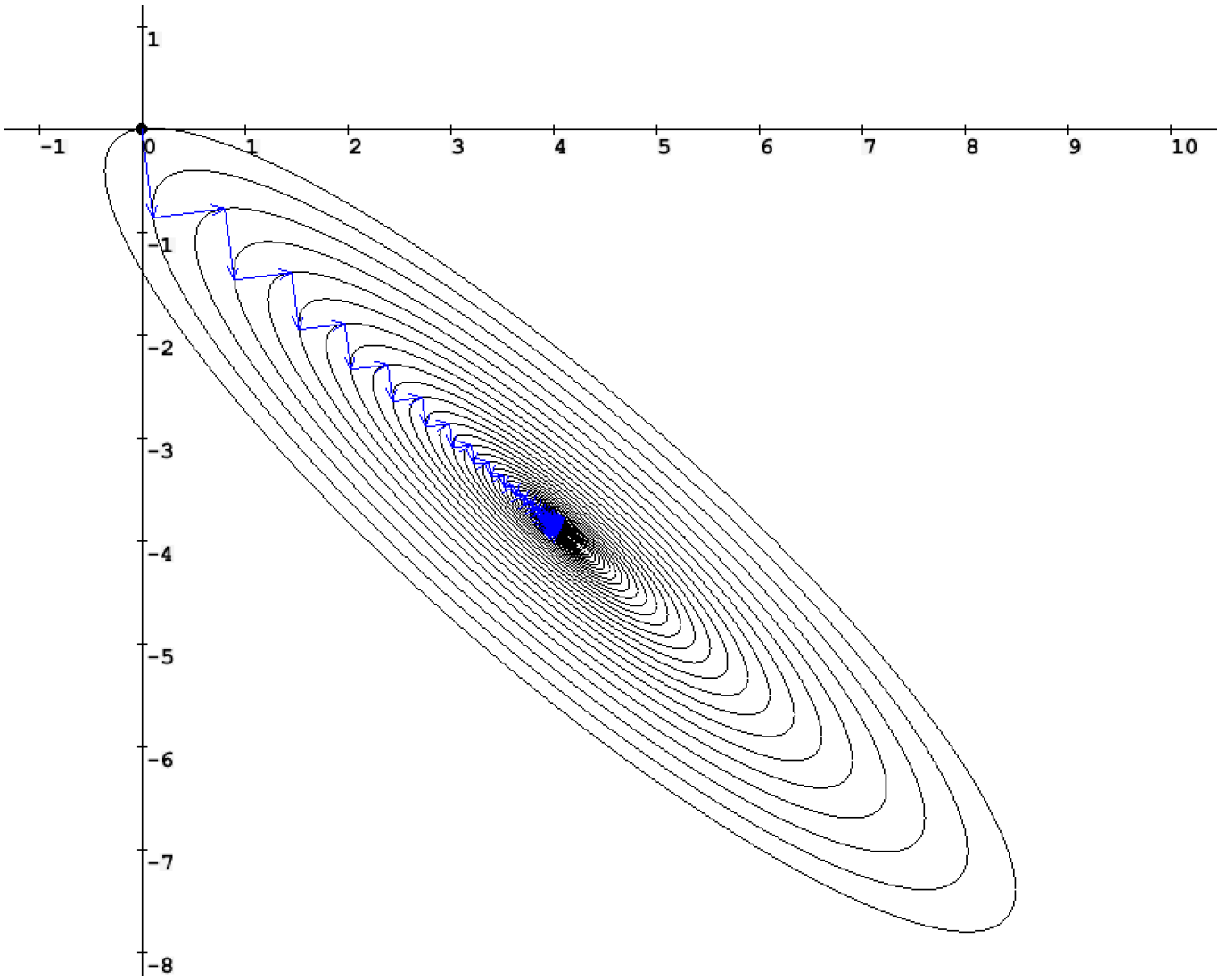


График 4 Function1, Fastest Gradient (Fibonacci)

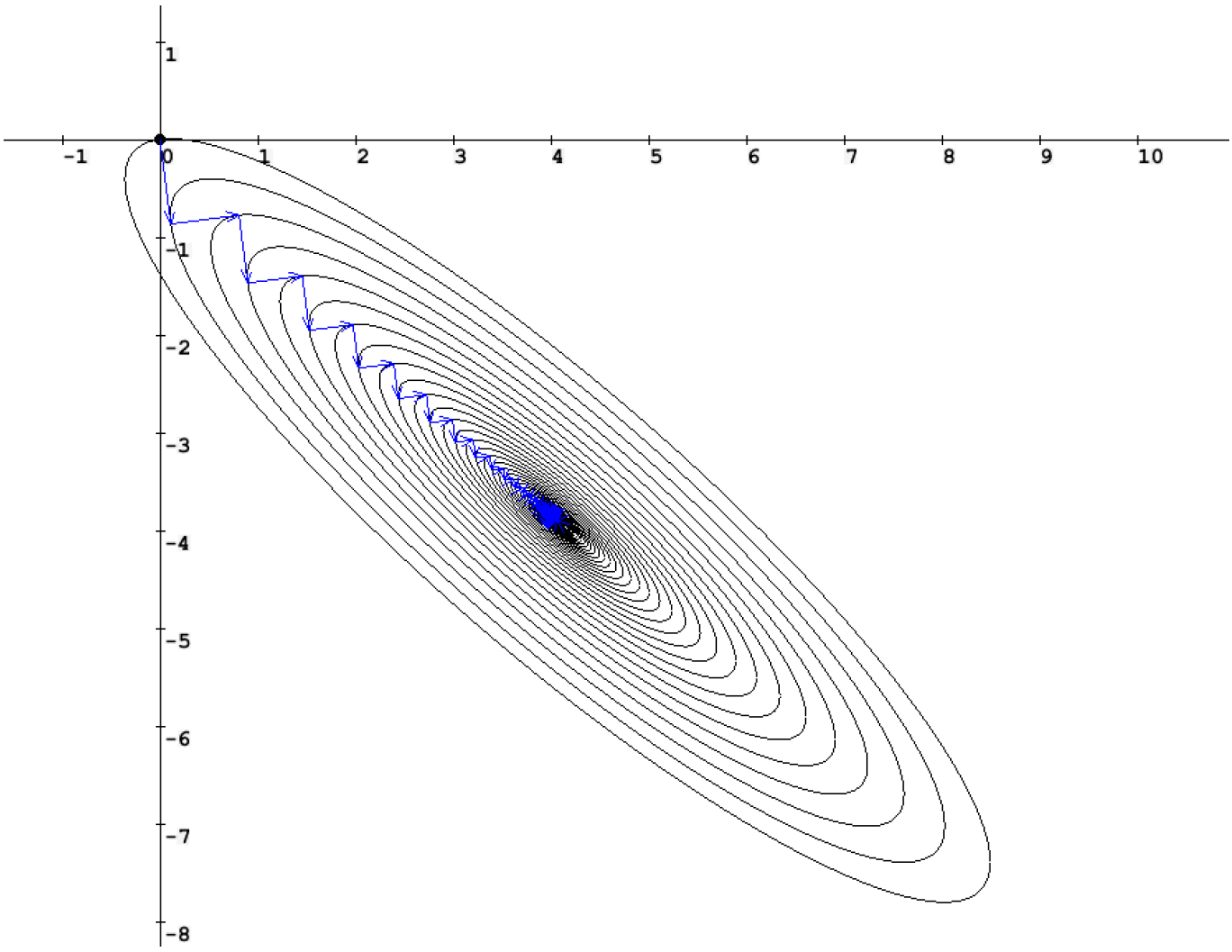


График 5 Function1, Fastest Gradient (Parabolas)

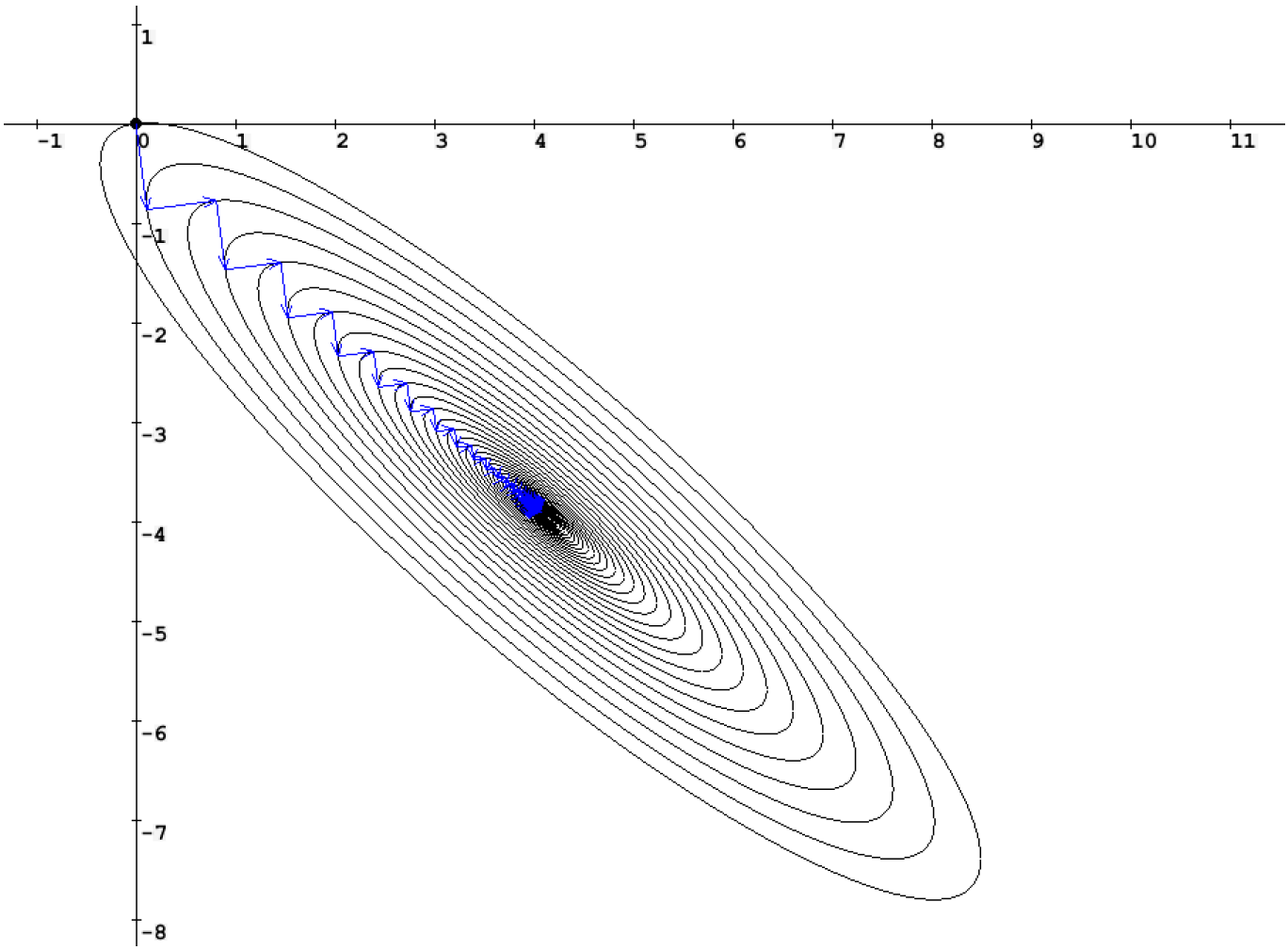


График 6 Function1, Fastest Gradient (Brent)

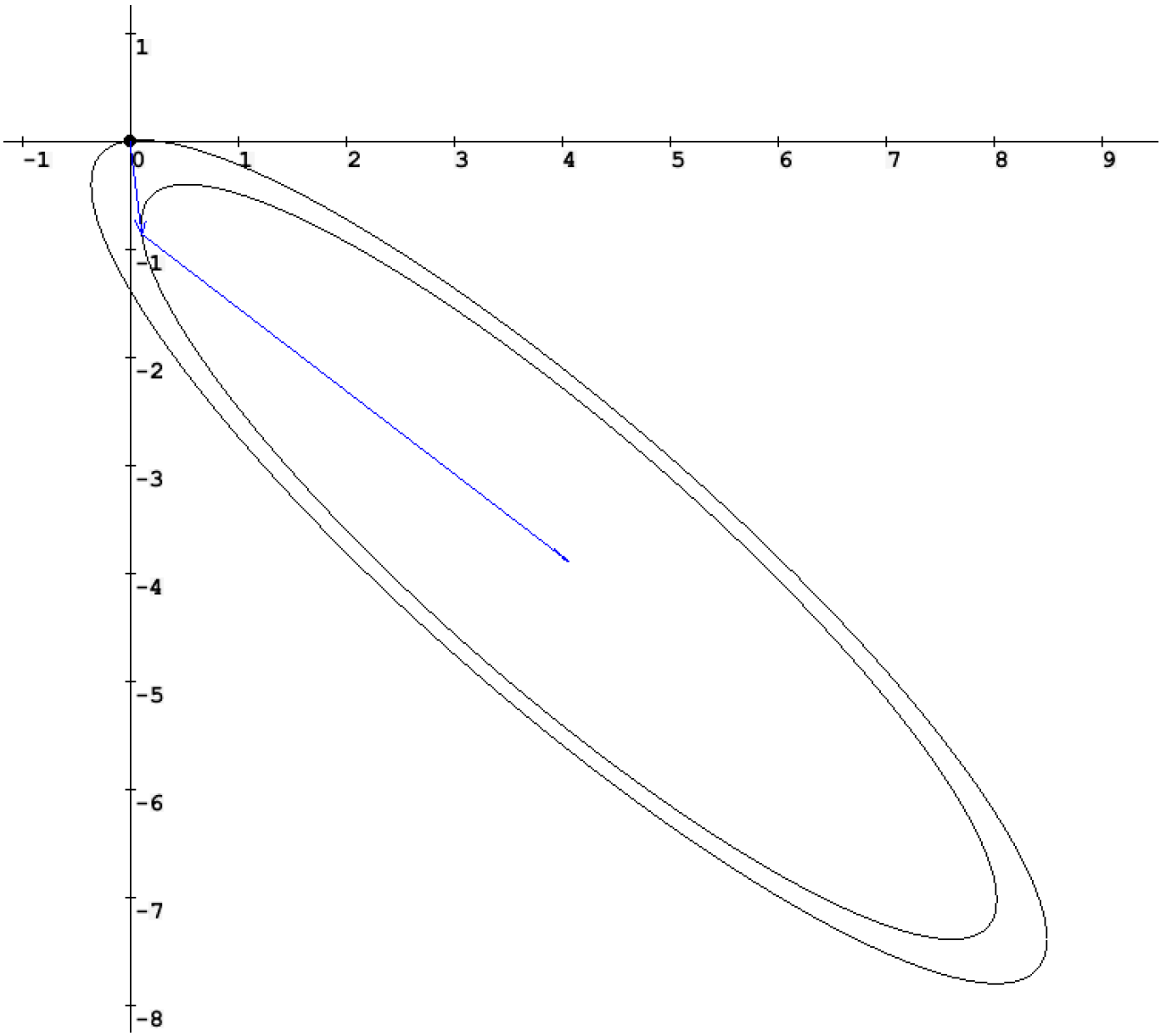


График 7 Function1, Conjugate Gradient

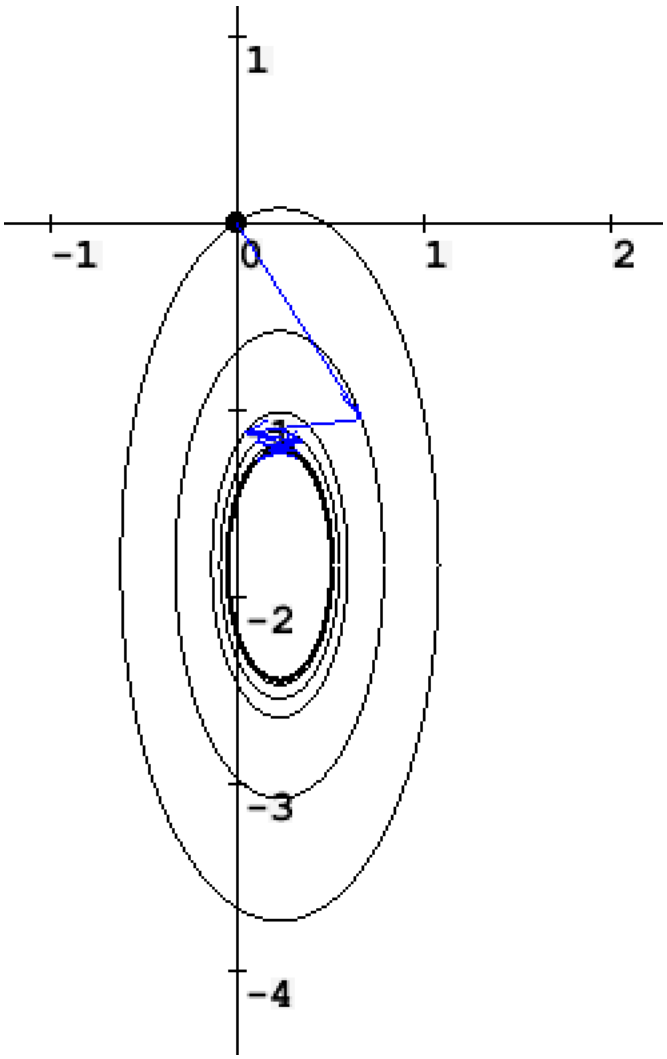


График 8 Function2, Gradient Descent

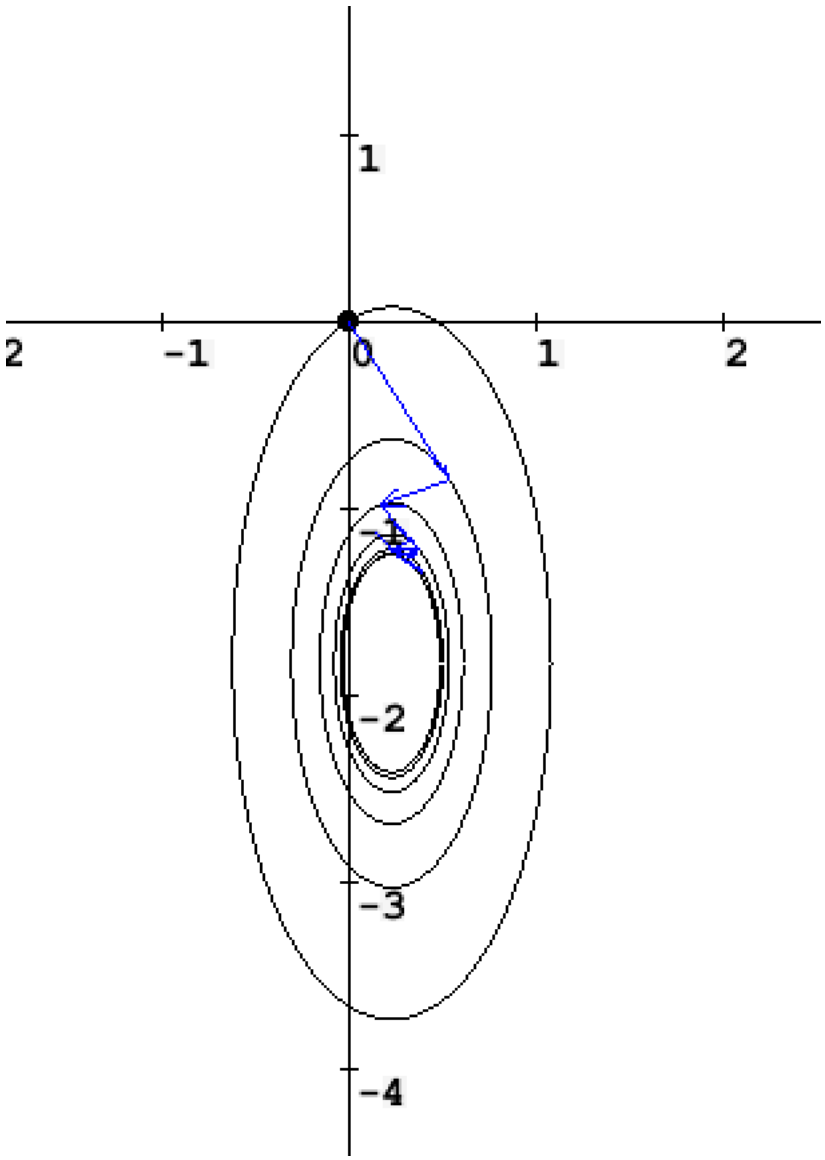


График 9 Function2, Fastest Gradient (Dichotomy)

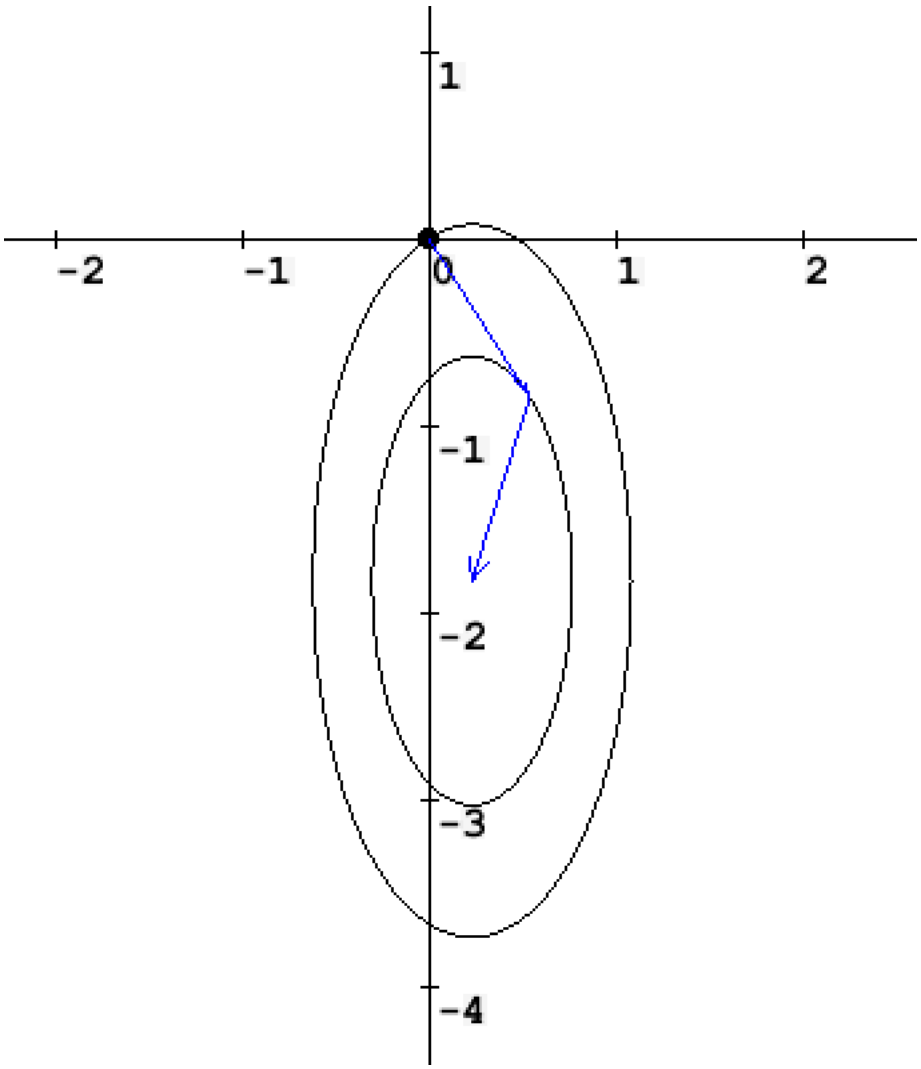
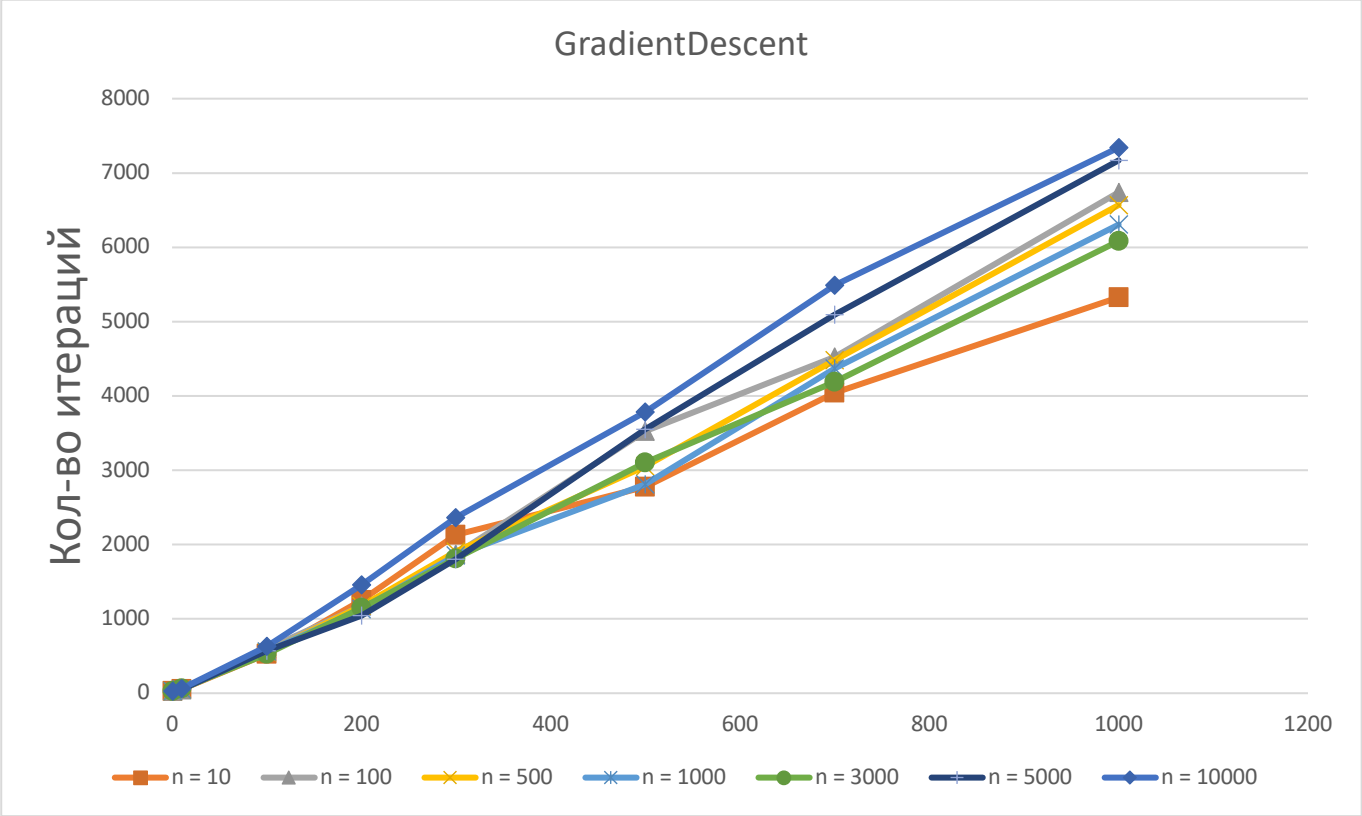


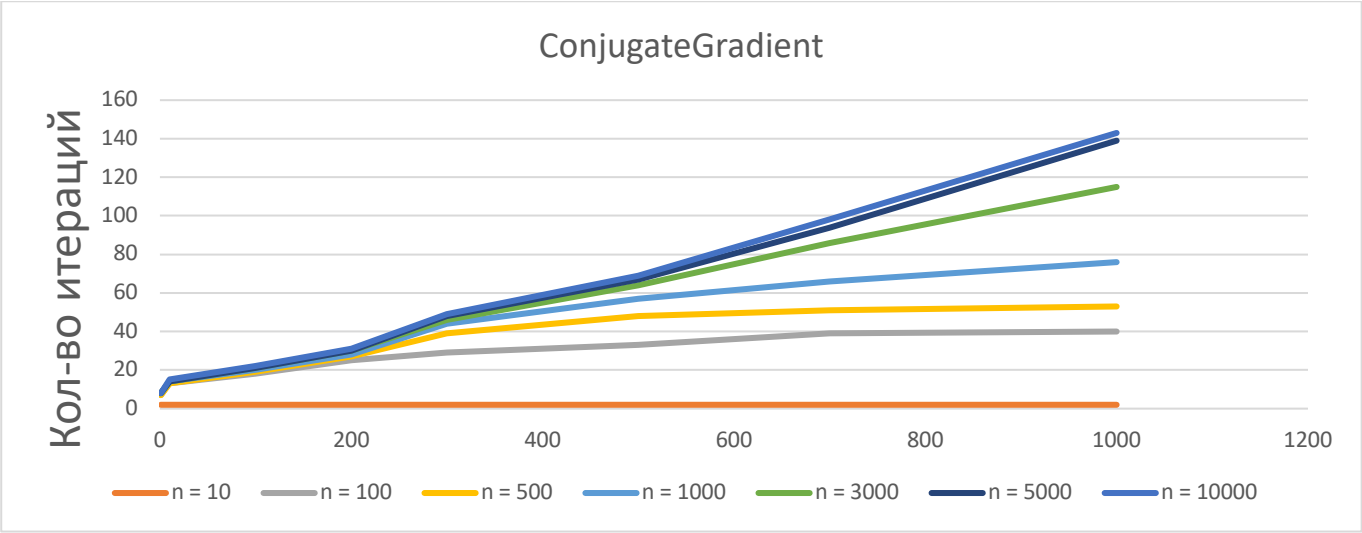
График 10 Function2, Conjugate Gradient

N	K	GradientDescent	ConjugateGradient	Dichotomy	Fibonacci	GoldenSection	Brent	Average	Difference
10									
	1	-3,027190032	-3,027190034	-3,027190024	-3,027190024	-3,027190024	-3,027190024	-3,027190027	9,92894E-09
	10	-2,587661254	-2,587661235	-2,58766125	-2,587661251	-2,58766125	-2,58766125	-2,587661248	1,95756E-08
	100	-2,824542585	-2,824542526	-2,824542583	-2,824542574	-2,824542583	-2,82454258	-2,824542572	5,91474E-08
	200	-3,503611893	-3,50361142	-3,503611872	-3,503611892	-3,503611871	-3,503611874	-3,503611804	4,72347E-07
	300	-5,819533361	-5,819533366	-5,819533341	-5,819533347	-5,819533348	-5,819533341	-5,819533351	2,48959E-08
	500	-1,892690928	-1,892689787	-1,892690907	-1,892690914	-1,892690909	-1,89269091	-1,892690726	1,14097E-06
	700	-2,72277251	-2,72277251	-2,722772502	-2,722772507	-2,722772504	-2,722772503	-2,722772506	8,53136E-09
	1000	-1,264643655	-1,264643655	-1,264643649	-1,264643648	-1,264643649	-1,264643649	-1,264643651	6,4408E-09
	2000	-2,46217564	-2,462175528	-2,462175625	-2,462175635	-2,462175626	-2,462175626	-2,462175613	1,12481E-07
100									
	1	-43,64197506	-43,64197507	-43,64197506	-43,64197506	-43,64197506	-43,64197506	-43,64197507	1,03689E-08
	10	-23,94416906	-23,94416894	-23,94416906	-23,94416906	-23,94416906	-23,94416906	-23,94416904	1,14715E-07
	100	-53,6200182	-53,61999942	-53,62001817	-53,62001817	-53,62001817	-53,62001817	-53,62001505	1,87732E-05
	200	-56,56985107	-56,56983786	-56,56985101	-56,56985102	-56,56985102	-56,56985101	-56,56984883	1,32158E-05
	300	-27,52472811	-27,52472748	-27,5247281	-27,5247281	-27,5247281	-27,52472811	-27,524728	6,35216E-07
	500	-38,58095402	-38,58094073	-38,580954	-38,580954	-38,580954	-38,58095401	-38,58095179	1,32887E-05
	700	-70,88807676	-70,88803235	-70,88807666	-70,88807671	-70,88807667	-70,88807666	-70,8880693	4,44138E-05
	1000	-16,28958656	-16,28958656	-16,28958655	-16,28958655	-16,28958655	-16,28958655	-16,28958655	4,77763E-09
	2000	-22,76479066	-22,7647906	-22,76479066	-22,76479066	-22,76479066	-22,76479066	-22,76479065	6,05629E-08
500									
	1	-124,2752646	-124,2752646	-124,2752646	-124,2752646	-124,2752646	-124,2752646	-124,2752646	5,69844E-09
	10	-123,264408	-123,2644075	-123,264408	-123,264408	-123,264408	-123,264408	-123,2644079	5,61585E-07
	100	-296,6753737	-296,6753459	-296,6753736	-296,6753736	-296,6753736	-296,6753736	-296,675369	2,78016E-05
	200	-174,2749727	-174,2749686	-174,2749727	-174,2749727	-174,2749727	-174,2749727	-174,274972	4,14737E-06
	300	-145,8647131	-145,8647112	-145,8647131	-145,8647131	-145,8647131	-145,8647131	-145,8647128	1,89827E-06
	500	-120,3362787	-120,3362787	-120,3362787	-120,3362787	-120,3362787	-120,3362787	-120,3362787	1,20525E-08
	700	-144,7620089	-144,7620059	-144,7620089	-144,7620089	-144,7620089	-144,7620089	-144,7620084	2,99266E-06
	1000	-101,9304846	-101,9304846	-101,9304846	-101,9304846	-101,9304846	-101,9304846	-101,9304846	4,95258E-09
	2000	-184,8487889	-184,8487836	-184,8487889	-184,8487889	-184,8487889	-184,8487889	-184,848788	5,3152E-06
1000									
	1	-5432,567511	-5432,567511	-5432,567511	-5432,567511	-5432,567511	-5432,567511	-5432,567511	1,69137E-07
	10	-435,3002955	-435,3002862	-435,3002955	-435,3002955	-435,3002955	-435,3002955	-435,300294	9,28274E-06
	100	-199,9381372	-199,9381371	-199,9381372	-199,9381372	-199,9381372	-199,9381372	-199,9381372	1,01043E-07
	200	-609,8892364	-609,8892057	-609,8892363	-609,8892363	-609,8892363	-609,8892363	-609,8892312	3,06485E-05
	300	-219,2726743	-219,2726743	-219,2726743	-219,2726743	-219,2726743	-219,2726743	-219,2726743	3,38274E-08
	500	-183,177755	-183,177755	-183,177755	-183,177755	-183,177755	-183,177755	-183,177755	2,29155E-08
	700	-269,349023	-269,3490222	-269,349023	-269,349023	-269,349023	-269,349023	-269,3490228	8,41231E-07
	1000	-184,1340802	-184,1340802	-184,1340802	-184,1340802	-184,1340802	-184,1340802	-184,1340802	5,09269E-09
	2000	-687,1101447	-687,1101112	-687,1101446	-687,1101447	-687,1101446	-687,1101446	-687,1101391	3,34777E-05
3000									
	1	-517,167577	-517,167577	-517,167577	-517,167577	-517,167577	-517,167577	-517,167577	3,81772E-09
	10	-552,6172736	-552,6172736	-552,6172736	-552,6172736	-552,6172736	-552,6172736	-552,6172736	1,08272E-08
	100	-593,6889011	-593,688901	-593,6889011	-593,6889011	-593,6889011	-593,6889011	-593,6889011	1,04184E-08
	200	-872,389338	-872,3893371	-872,389338	-872,389338	-872,389338	-872,389338	-872,3893378	8,24026E-07
	300	-936,78191	-936,7819091	-936,78191	-936,78191	-936,7819099	-936,78191	-936,7819098	8,20012E-07
	500	-1561,904122	-1561,904111	-1561,904122	-1561,904122	-1561,904122	-1561,904122	-1561,90412	1,08704E-05
	700	-1074,858285	-1074,858284	-1074,858285	-1074,858285	-1074,858285	-1074,858285	-1074,858285	1,34027E-06
	1000	-523,2662774	-523,2662774	-523,2662774	-523,2662774	-523,2662774	-523,2662774	-523,2662774	4,18663E-09
	2000	-2945,672137	-2945,671847	-2945,672137	-2945,672137	-2945,672137	-2945,672137	-2945,672088	0,000290526
5000									
	1	-1628,772998	-1628,772998	-1628,772998	-1628,772998	-1628,772998	-1628,772998	-1628,772998	9,20886E-09
	10	-1460,045864	-1460,045863	-1460,045864	-1460,045864	-1460,045864	-1460,045864	-1460,045864	1,15089E-06
	100	-929,5716769	-929,5716769	-929,5716769	-929,5716769	-929,5716769	-929,5716769	-929,5716769	4,77974E-09
	200	-962,3624939	-962,3624939	-962,3624939	-962,3624939	-962,3624939	-962,3624939	-962,3624939	5,02644E-09
	300	-2221,740421	-2221,740415	-2221,740421	-2221,740421	-2221,740421	-2221,740421	-2221,74042	5,79558E-06
	500	-1291,2482	-1291,2482	-1291,2482	-1291,2482	-1291,2482	-1291,2482	-1291,2482	4,78244E-08
	700	-952,1650899	-952,1650899	-952,1650899	-952,1650899	-952,1650899	-952,1650899	-952,1650899	5,2778E-09
	1000	-1498,096266	-1498,096266	-1498,096266	-1498,096266	-1498,096266	-1498,096266	-1498,096266	3,3881E-07
	2000	-1390,338988	-1390,338987	-1390,338988	-1390,338988	-1390,338988	-1390,338988	-1390,338988	7,18838E-07
10000									
	1	-2360,491308	-2360,491308	-2360,491308	-2360,491308	-2360,491308	-2360,491308	-2360,491308	5,22095E-09
	10	-1958,630701	-1958,630701	-1958,630701	-1958,630701	-1958,630701	-1958,630701	-1958,630701	9,87325E-09
	100	-2598,833818	-2598,833817	-2598,833818	-2598,833818	-2598,833818	-2598,833818	-2598,833817	5,60271E-08
	200	-3805,483736	-3805,483734	-3805,483736	-3805,483736	-3805,483736	-3805,483736	-3805,483736	1,59711E-06
	300	-3077,251876	-3077,251874	-3077,251876	-3077,251876	-3077,251876	-3077,251876	-3077,251876	1,35788E-06
	500	-3231,057176	-3231,057176	-3231,057176	-3231,057176	-3231,057176	-3231,057176	-3231,057176	2,22487E-07
	700	-2455,637988	-2455,637988	-2455,637988	-2455,637988	-2455,637988	-2455,637988	-2455,637988	1,0542E-08
	1000	-1923,047976	-1923,047976	-1923,047976	-1923,047976	-1923,047976	-1923,047976	-1923,047976	5,26984E-09
	2000	-2339,187717	-2339,187717	-2339,187717	-2339,187717	-2339,187717	-2339,187717	-2339,187717	1,28206E-07

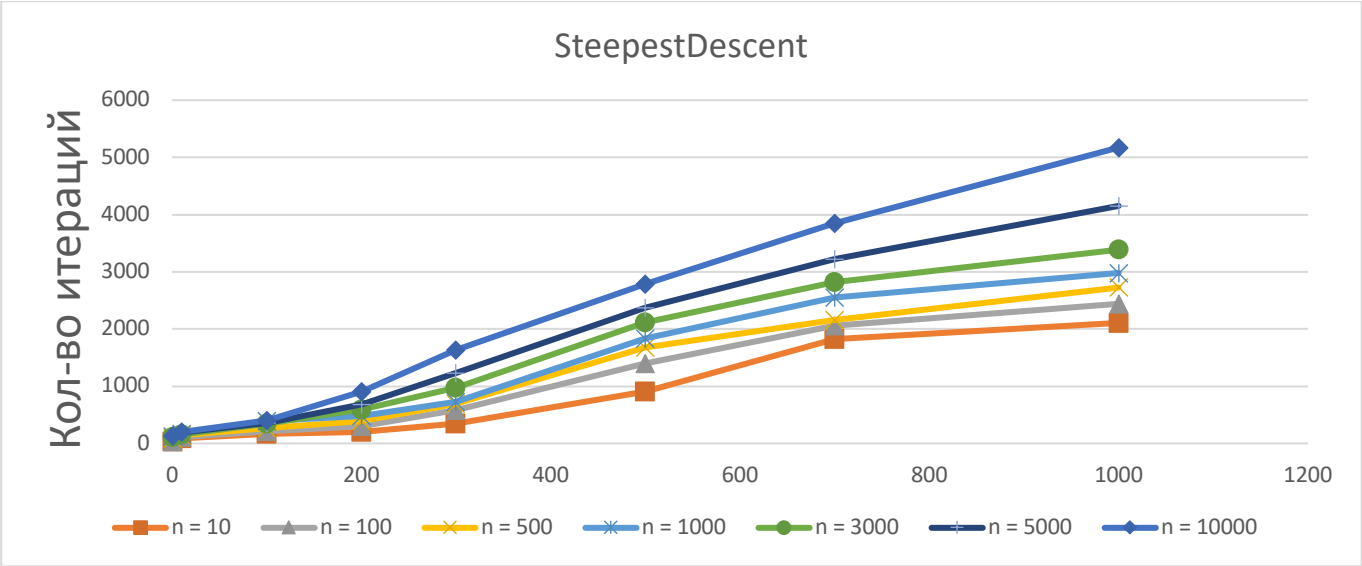
Исследование зависимости числа итераций от числа обусловленности и размерности пространства



Можно отметить, что для фиксированного числа обусловленности количество итераций постоянно.



Как видно из графиков кол-во итераций не превышает размерности.



С увеличением n возрастает количество итераций.

Method	n
Dichotomy	434
Fibonacci	651
GoldenSection	558
Brent	527

Таблица 1 Кол-во итераций одномерных методов оптимизаций при n = 10, k = 1

Method	n
Dichotomy	215566
Fibonacci	162108
GoldenSection	335510
Brent	500346

Таблица 2 Кол-во итераций одномерных методов оптимизаций при n = 10000, k = 2000

Важно заметить:

- кол-во итераций у различных одномерных методов примерно одного порядка
- решение задачи одномерной оптимизации производит в разы больше итераций чем основной алгоритм.

Сравнение методов

В метод градиентного спуска совершает наибольшее количество итераций, что отрицательно сказывается на его производительности.
Метод наискорейшего спуска совершает меньше действий, чем метод градиентного спуска, но больше, чем метод сопряженных градиентов. Стоит также учитывать кол-во итераций методов одномерной оптимизации. Было выяснено, что если за кол-во итераций метода наискорейшего спуска брать сумму итераций самого метода и методов одномерной оптимизации, то получаем, что метод наискорейшего спуска самый медленный. При всем при этом метод наискорейшего спуска показывает наилучшую точность вычислений.
Выбор способа решения задачи одномерной оптимизации не оказывает сильного влияния на кол-во итераций, а значит не влияет на производительность.
Метод сопряжённых градиентов показал себя самым стабильным и наиболее быстрым методом по количеству итераций.

Исходный код: https://github.com/AntonAsmirko/Optimization-Methods