МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

Санкт - Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

Лабораторная работа №2

По дисциплине «Прикладная математика»

Выполнил студент группы №М32081
Предыбайло Влада,
Куксенко Кристина,
Алексеева Милена

Проверил
Москаленко Мария Александровна

САНКТ-ПЕТЕРБУРГ

2021

https://github.com/is-y23/applied-math-team-45

Методы спуска для реализации

1) Метод наискорейшего спуска
2) Метод градиентного спуска
3) Метод сопряженных градиентов
4) Метод сопряженных направлений
5) Метод Ньютона

Методы поиска величины шага

1) Постоянная величина шага
2) Метод дробления шага
3) Метод золотого сечения
4) Метод Фибоначчи

Изначальные функции

1) Проанализировать работу алгоритмов на функции

$$f(x_1, x_2) = x_1^2 + x_1 x_2 + x_2^2 - 2x_1 + x_2.$$

При начальном приближении (0, 1) и точности 0,1

2) Проанализировать работу алгоритмов на функции

$$f(x, y) = \frac{x^2}{5} + \frac{y^2}{5} - sin(x) + sin(y)$$

При начальном приближении (-1, 1) и точности 0,1

Метод наискорейшего спуска

Метод наискорейшего спуска - это итерационный численный метод (первого порядка) решения оптимизационных задач, который позволяет определить экстремум (минимум или максимум) целевой функции. В текущей точке вычисляется Grad F(X), и затем в

направлении антиградиента ищется Min F(X). Практически это может быть осуществлено любым методом одномерной оптимизации, наиболее часто используется сканирование до первого локального минимума по направлению анти Grad F(X). В данном методе для поиска Min F(x), мы реализовали две вариации, воспользовавшись методами: золотого сечения и Фибоначчи

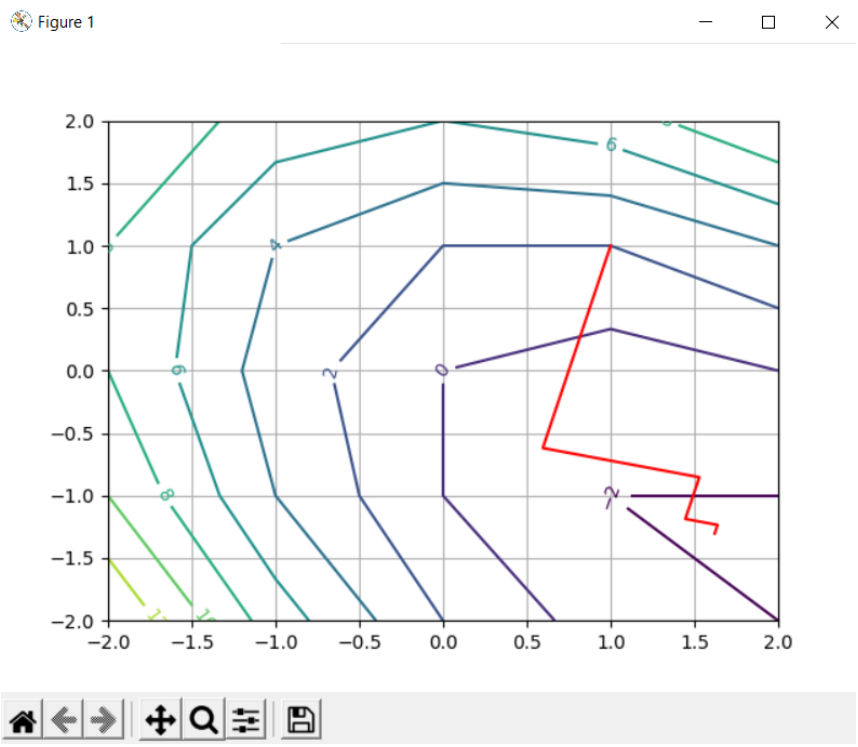
Золотое сечение:

Step	x	y
1	0,595238099	-0,619047606
2	1,529304036	-0,852564113
3	1,445905294	-1,186159072
4	1,638363918	-1,234273747
5	1,62118012	-1,303008967
	Step count = 5	

Фибоначчи:

Step	x	y
1	0,595238097	-0,619047613
2	1,52930403	-0,852564107
3	1,44590529	-1,186159069
4	1,638363933	-1,234273749
5	1,621180126	-1,303008965
	Step count = 5	

Поточечная эмуляция золотого сечения и Фибоначчи:



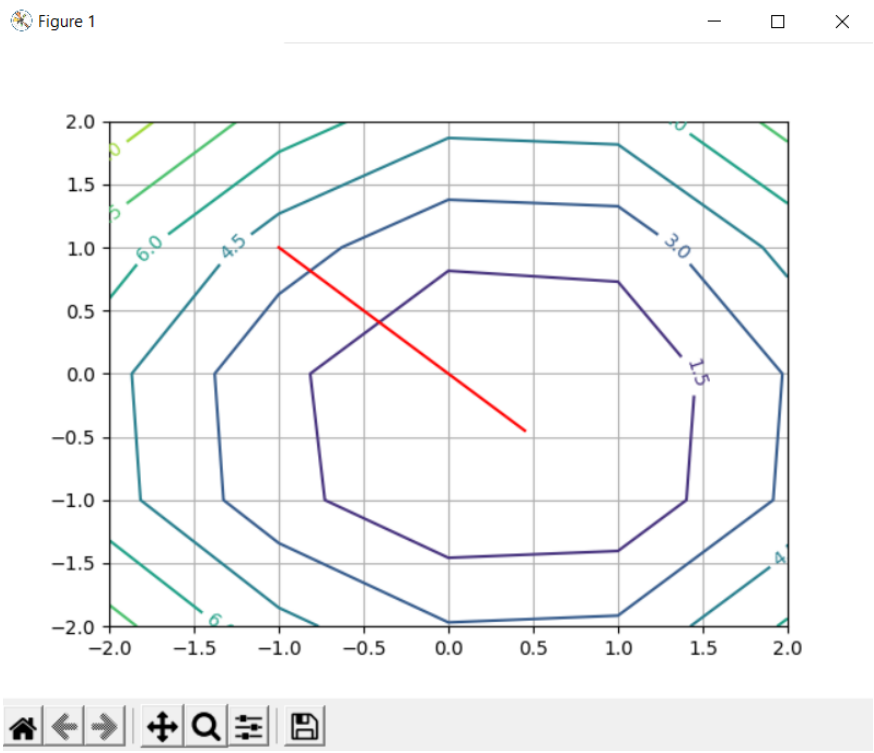
Результаты для второй функции:

Фибоначчи

Step	x	y
1	0,450183606	-0,450183606
	Step count = 1	

Золотое сечение

Step	x	y
1	0,450183606	-0,450183606
	Step count = 1	



Получили сходимость за 5 и 1 шаг с точностью 0.1. В случае функции двух переменных данный метод имеет следующую геометрическую интерпретацию: направление движения из точки x_k касается линии уровня в точке x_{k+1}. Траектория спуска зигзагообразная, причем соседние звенья зигзага ортогональны друг другу.

Градиентный спуск с постоянной величиной шага

Известно, что направление наибольшего возрастания функции двух переменных u=f(x, y) характеризуется ее градиентом:

Следовательно, направление, противоположное градиенту, укажет путь, наибольшего убывания функции.

Сущность метода: В качестве начала итераций выбирается произвольная точка(опорная точка). Величина шага задается пользователем и остается постоянной до тех пор, пока функция убывает. Если это условие не выполняется, то производится коррекция длины шага. Процесс завершается при достижении заданного уровня точности

Поиск каждой новой точки состоит из двух этапов:

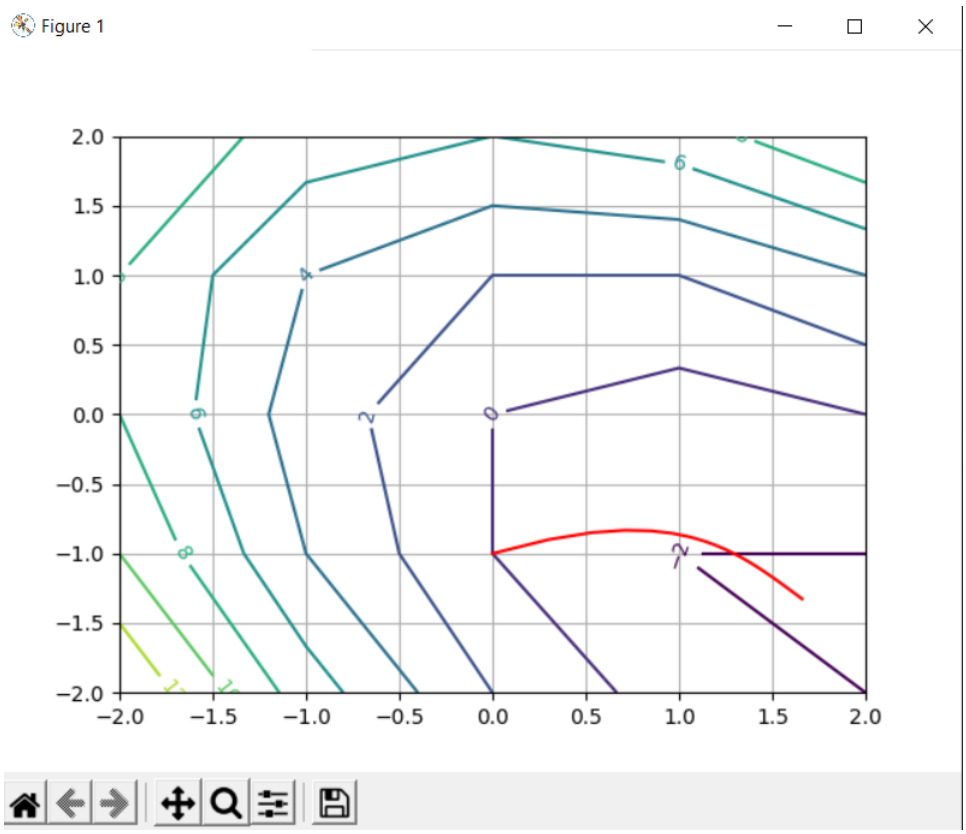
Оценка градиента F(X) путем вычисления частных производных от F(x) По каждой переменной Xj; Передвижение по рабочему шагу по всем переменным одновременно в направлении антиградиента .

Метод градиента в чистом виде формирует спуск по переменным как функцию от градиента F(x) в текущей точке поиска. Простейший алгоритм поиска Min F(X) записывается в векторной форме следующим образом:

$$x^{i+1} = x^i - h \cdot gradf(x^i),$$

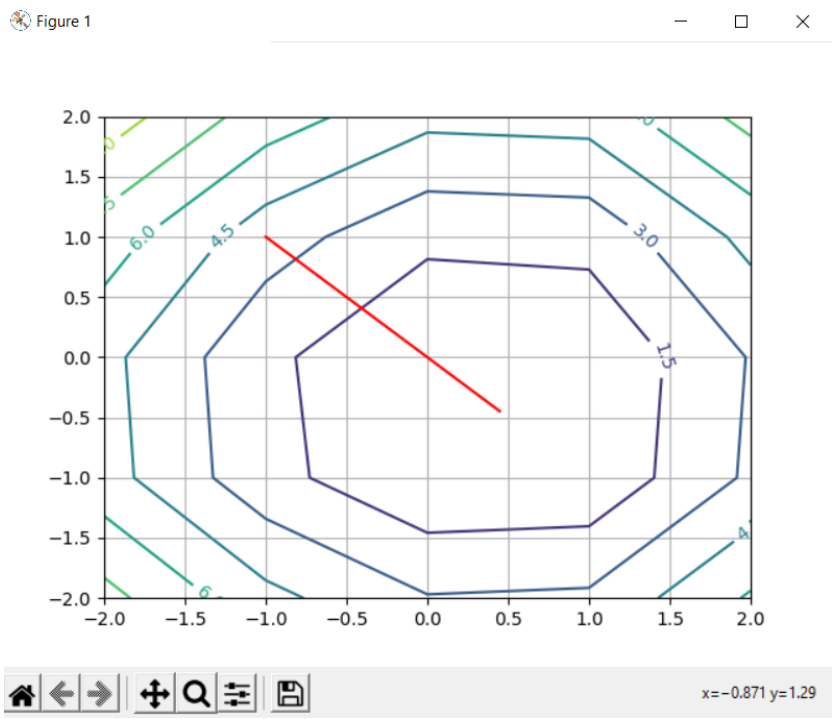
Step	x	y
1	0,3	-0,9
2	0,53	-0,85
3	0,709	-0,833
4	0,8505	-0,8373
5	0,96413	-0,85489
...	...	...
44	1,6558912574147	-1,322558215257
45	1,6569688274575	-1,323635697947
46	1,6579386317606	-1,324605441103
47	1,6588114495188	-1,325478216059
48	1,6595969812209	-1,326263717799
	Step count = 48	
	Size of step = 0.1	

Поточечная эмуляция:



Результаты для второй функции:

Step	x	y
1	-0,745969769	0,745969769
2	-0,523332807	0,523332807
3	-0,33205041	0,33205041
4	-0,171102735	0,171102735
5	-0,038342427	0,038342427
...		
97	0,450183611	-0,450183611
98	0,450183611	-0,450183611
99	0,450183611	-0,450183611
100	0,450183611	-0,450183611
101	0,450183611	-0,450183611
	Step count = 101	



С уменьшением точности увеличилось количество итераций, но и увеличилась точность решения. И обратное – с увеличением точности количество итераций уменьшается и точность тоже.

Градиентный спуск с дроблением шага

Градиентный метод с дроблением шага линейно сходится. Однако, у данного метода есть и минусы. Умножение числа на заранее выбранное бета принадлежащее (0,1) несет в себе затраты на вычисления на каждом шаге.

Пусть F(X) выпуклая дифференцируемая во всем N-Мерном пространстве ε_n Функция и требуется найти её точку минимума X*. Выбрав произвольное начальное приближение $x^{(0)} \in \varepsilon_n$, построим последовательность

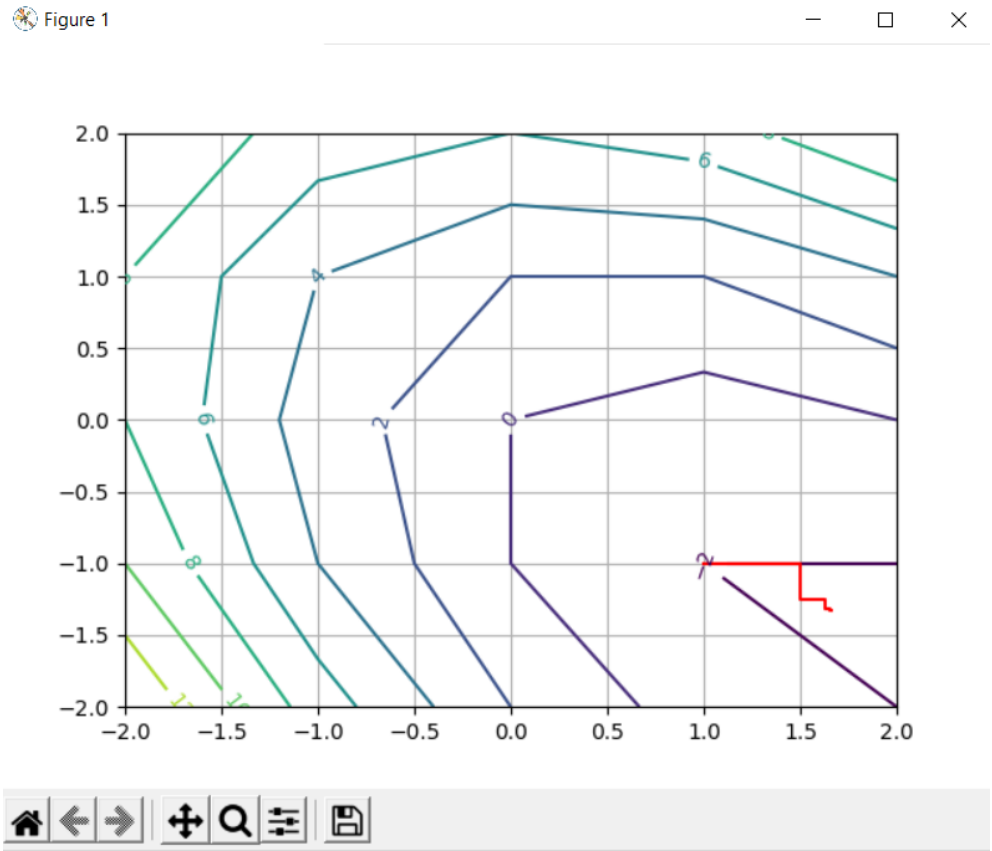
$$x^{(k+1)} = {}^{(k)} - h_k f'(x^{(k)}),\ k = 0,1,\ldots,$$

Где величины Hk (параметрические шаги) выбираются достаточно малыми для того, чтобы выполнялось условие

$$f(x^{\lfloor k+1 \rfloor}) = f(x^{\lfloor k \rfloor} - \lambda^{\lfloor k \rfloor} f'(x^{\lfloor k \rfloor})) \le f(x^{\lfloor k \rfloor}) - \epsilon \lambda^{\lfloor k \rfloor} \| f'(x^{\lfloor k \rfloor}) \|^2.$$

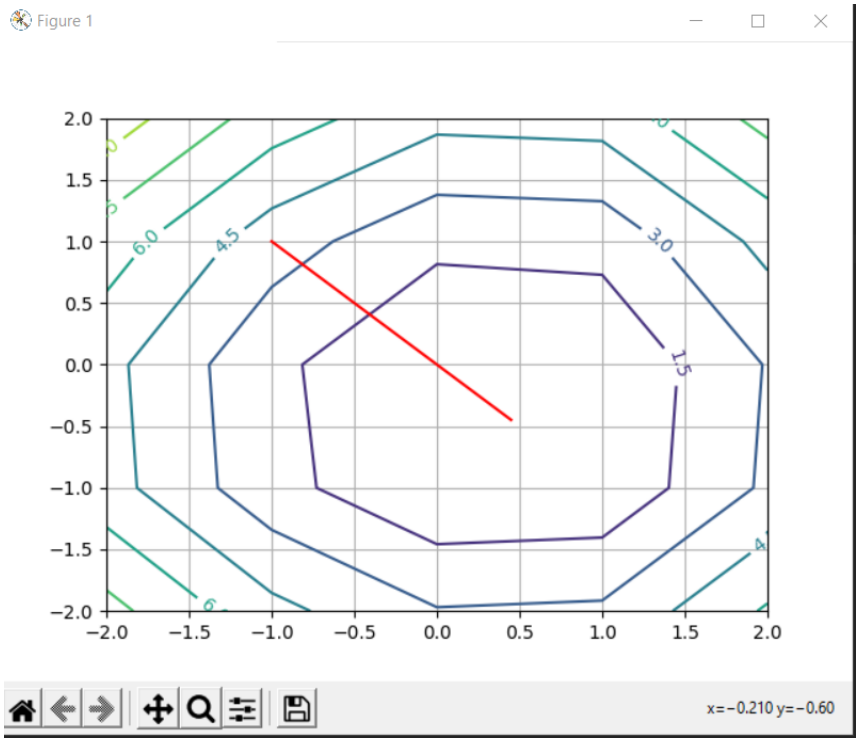
Step	x	y
1	1	-1
2	1,5	-1
3	1,5	-1,25
4	1,625	-1,25
5	1,625	-1,3125
6	1,65625	-1,3125
7	1,65625	-1,328125
8	1,6640625	-1,328125
	Step count = 8	

Поточечная эмуляция:



Результаты для второй функции:

Step	x	y
1	-1	1
2	0,270151153	-0,270151153
3	0,270151153	-0,270151153
...		
7	0,445592106	-0,445592106
8	0,448384961	-0,448384961
	Step count = 8	



Метод сопряженных градиентов

Достоинство этого метода в том, что он решает квадратичную задачу оптимизации за конечное число шагов. В этом методе мы работаем через антиградиент. Также, нужно учитывать при выборе направления предыдущие градиенты.

На многих других задачах метод сопряжённого градиента также превосходит метод градиентного спуска. Сходимость метода градиентов существенно зависит от того, насколько точно решается задача одномерной оптимизации $F(x_{k-1}+\alpha_k p_k) \to \min\limits_{\alpha_k \geq 0}$. Возможные зацикливания метода устраняются с помощью обновлений. Тем не

менее, если метод попадёт в локальный минимум функции, скорее всего, ему не удастся из него выбраться.

.В общем случае процесс нахождения минимума функции является итерационной процедурой, которая записывается в векторной форме следующим образом:

$$X_{k+1} = X_k \pm \lambda_k \cdot P_k$$

Pk - единичный вектор сопряженных направлений, который определяется по формуле:

$$P_0 = \nabla f(X_0)$$

$$P_k = \nabla f(X_k) + \beta_k \cdot P_{k-1}, \quad k = 1, 2, \ldots$$

В свою очередь весовой коэффициент определяется по формуле:

$$\beta_k = \frac{\left\|\nabla f(x_1, x_2, \cdots, x_n)_k\right\|^2}{\left\|\nabla f(x_1, x_2, \cdots, x_n)_{k-1}\right\|^2} = \frac{\sum_{i=1}^{n}\left(\frac{\partial f(x_1, x_2, \cdots, x_n)}{\partial x_i}\right)_k^2}{\sum_{i=1}^{n}\left(\frac{\partial f(x_1, x_2, \cdots, x_n)}{\partial x_i}\right)_{k-1}^2}$$

Таким образом, метод сопряженных градиентов формирует направление поиска к оптимальному значению используя информацию о поиске полученную на предыдущих этапах спуска.

В нашем случае метод обеспечил одну из наилучших сходимостей за 3 и 2 шага. Но для плохо обусловленных функций и больших точностей основная проблема заключается в том, что из-за накопления погрешностей может нарушаться ортогональность базисных векторов, что ухудшает сходимость

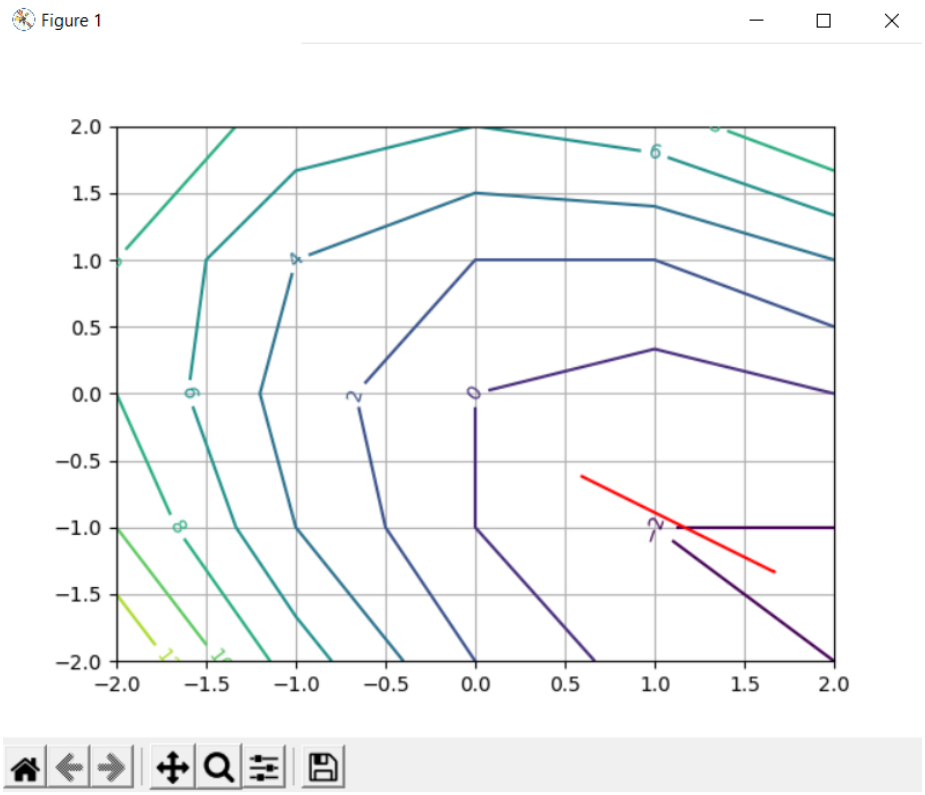
Фибоначчи:

Step	x	y
1	0,595238097	-0,61904761
2	1,666666668	-1,33333334
3	1,666666669	-1,33333334

Золотое сечение:

Step	x	y
1	0,595238099	-0,61904761
2	1,666666643	-1,33333333
3	1,666666657	-1,33333332

Поточечная эмуляция золотого сечения и Фибоначчи:



Результаты для второй функции:

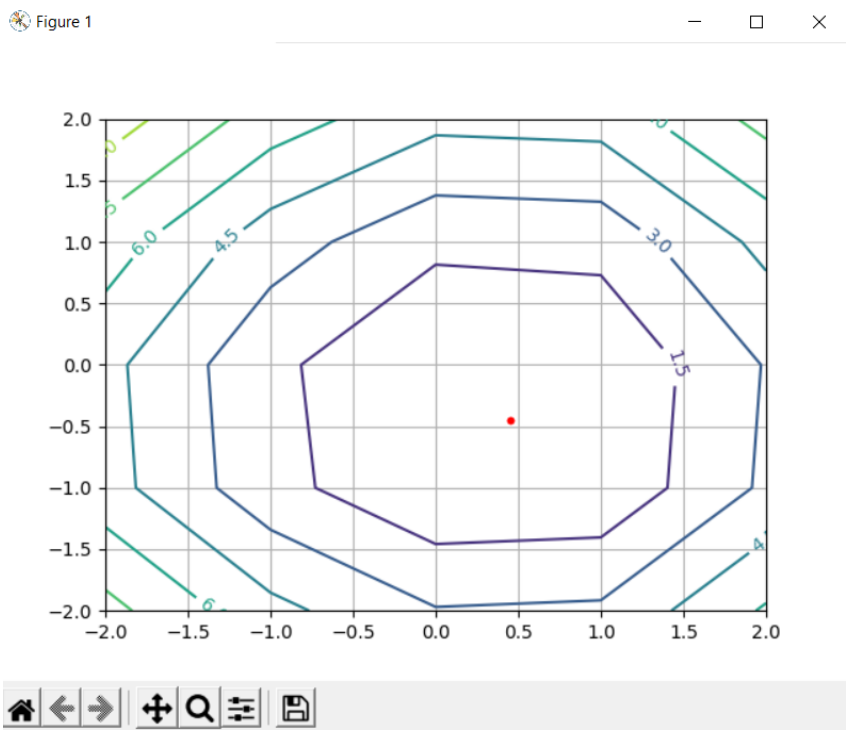
Золотое сечение

Step	x	y
1	0,450183604	-0,450183604
2	0,450183607	-0,450183607
	Step count = 2	

Фибоначчи

Step	x	y
1	0,450183606	-0,450183606
2	0,450183613	-0,450183613
	Step count = 2	

Поточечная эмуляция золотого сечения и Фибоначчи:



Методы золотого сечения и Фибоначчи.

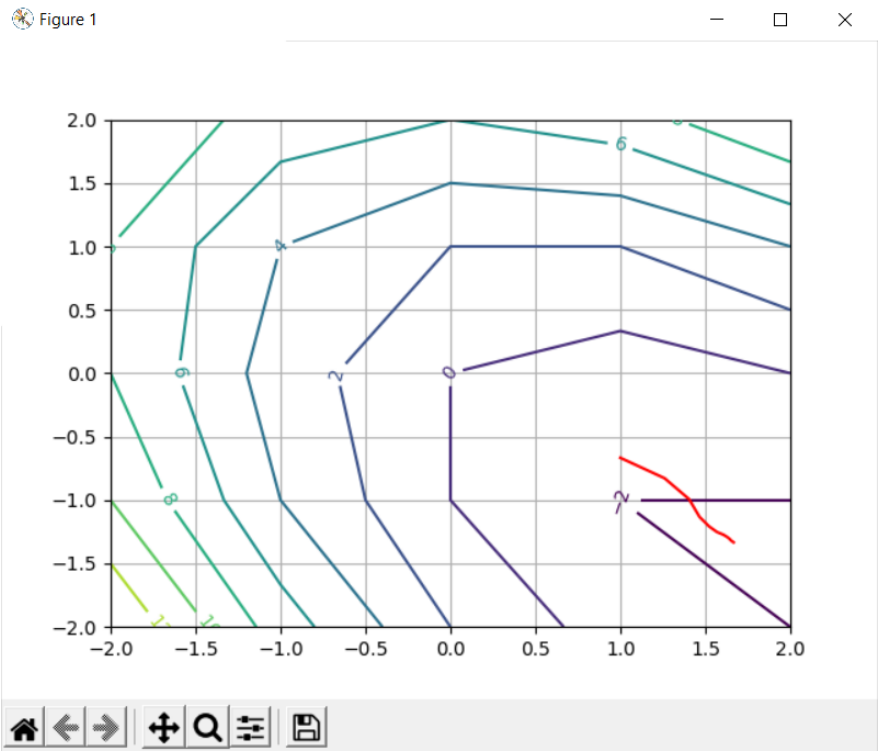
Эти два метода могут быть применены для решения задачи с помощью метода наискорейшего спуск или сопряженных градиентов. Из плюсов данного метода можем выделить то, что они сходятся быстрее всего. Однако надо решать еще и нулевые методы, в какой либо плоскости, что усложняет процесс.

Метод сопряженных направлений

В методе сопряженных направлений используется факт, что минимум квадратичной функции может быть найден не более чем за n шагов при условии, что поиск ведется вдоль сопряженных относительно матрицы Гессе направлений. Задается начальная точка и направления, совпадающие с координатами. Находится минимум f(x) при последовательном движении по (n + 1) направления с помощью одного из методов одномерной минимизации. При этом полученная ранее точка минимума берется в качестве исходной для поиска по совершенно нового направлению, а направление используется как при первом, так и последнем поиске. Находится новое направление поиска. Оно проходит через точки, полученные при последнем поиске. Направление заменяется сопряженным направлением, после чего повторяется поиск по (n+1) направлениям, уже не содержащим старого направления. Основная его идея заключается в том, что бы, приведя квадратичную функцию n переменных к виду суммы полных квадратов, был найден ее оптимум в результате n одномерных поисков по преобразованным координатным направлениям.

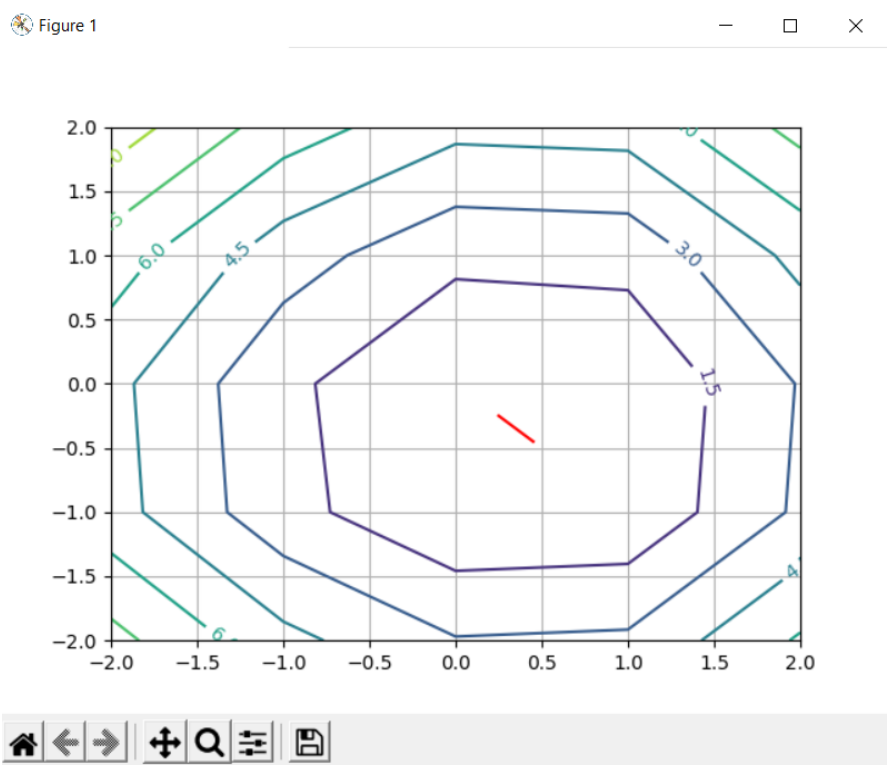
Step	x	y
1	1	-0,666666667
2	1,26010929	-0,827686703
3	1,465700652	-1,132367319
4	1,521956297	-1,207402765
5	1,606654625	-1,273321292
...	...	...
45	1,666666604	-1,333333271
46	1,666666639	-1,333333306
47	1,666666648	-1,333333315
48	1,666666658	-1,333333325
49	1,666666661	-1,333333328
	Step count = 49	

Поточечная эмуляция:



Результаты для второй функции:

Step	x	y
1	0,25	-0,25
2	0,342838786322	-0,342838786
3	0,415448799	-0,415448799
4	0,429994995	-0,429994995
5	0,443259137	-0,443259137
...		
33	0,45018355	-0,45018355
34	0,45018359	-0,45018359
35	0,450183599	-0,450183599
36	0,450183607	-0,450183607
37	0,450183609	-0,450183609
	Step count = 37	



Хотя подобное разложение и гарантирует хорошую сходимость, метод ведет себя гораздо хуже метода сопряженных градиентов, т. к. шаг не всегда разумно выбирается(например Гессиан не всегда положительно определен). Поэтому чаще всего целесообразно использовать его модификации с решением задач одномерной оптимизации, вместо оригинального подхода.

Метод Ньютона

При наличии хорошего приближения xk к корню $\bar{x}$ функции f(·) можно использовать метод Ньютона, называемый также методом линеаризации или методом касательных. Метод Ньютона основан на использовании квадратичной аппроксимации функции f в окрестности текущей точки xk:

$$f(\boldsymbol{x}_k + \boldsymbol{d}_k) \simeq f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^T \boldsymbol{d}_k + \frac{1}{2}\boldsymbol{d}_k^T \nabla^2 f(\boldsymbol{x}_k)\boldsymbol{d}_k \to \min_{\boldsymbol{d}_k}.$$

Здесь и далее будем обозначать gk = ∇ f(xk), Hk = ∇ 2f(xk).

Приравнивая к нулю градиент квадратичной аппроксимации, получаем:

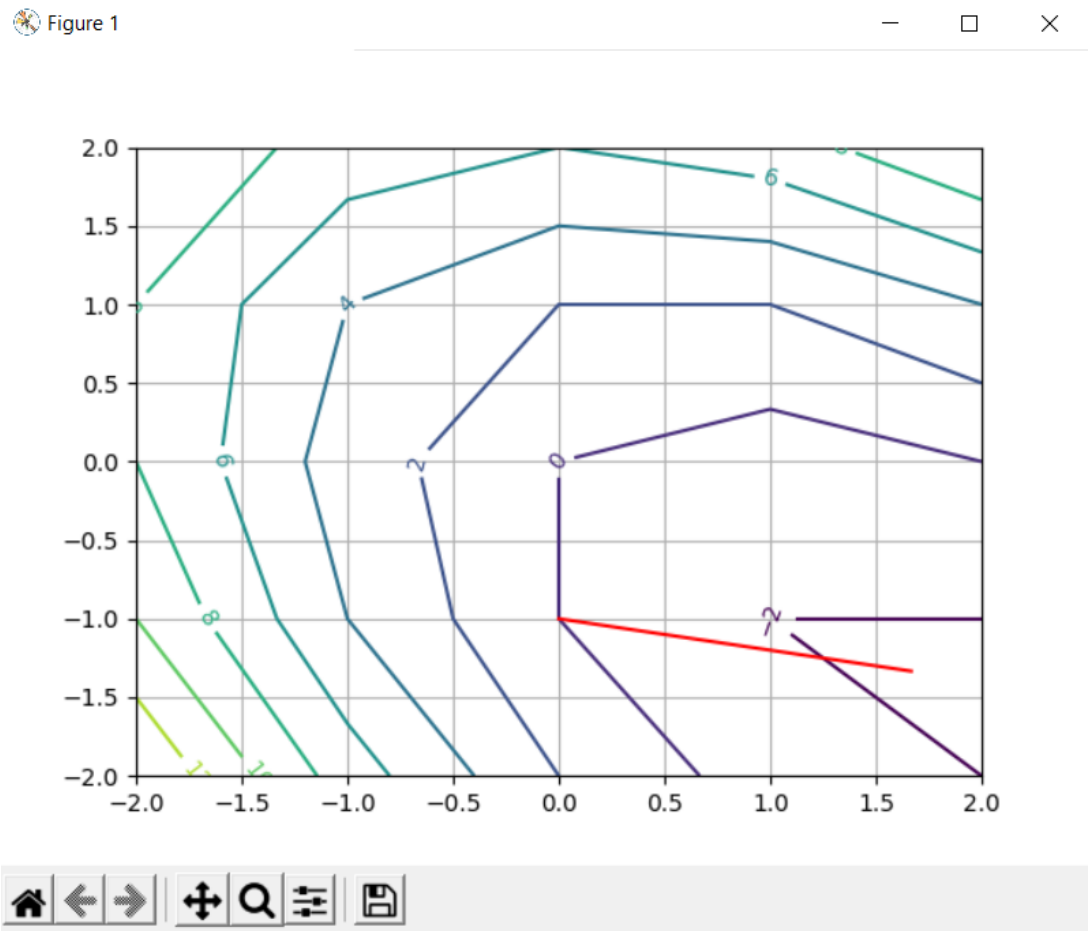
$$\nabla_{\boldsymbol{d}_k} = \boldsymbol{g}_k + H_k \boldsymbol{d}_k = 0 \;\Rightarrow\; H_k \boldsymbol{d}_k = -\boldsymbol{g}_k \;\Rightarrow\; \boldsymbol{d}_k = -H_k^{-1} \boldsymbol{g}_k.$$

Решение этого уравнения принимается за очередное приближение к искомому корню уравнения

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - H_k^{-1} \boldsymbol{g}_k.$$

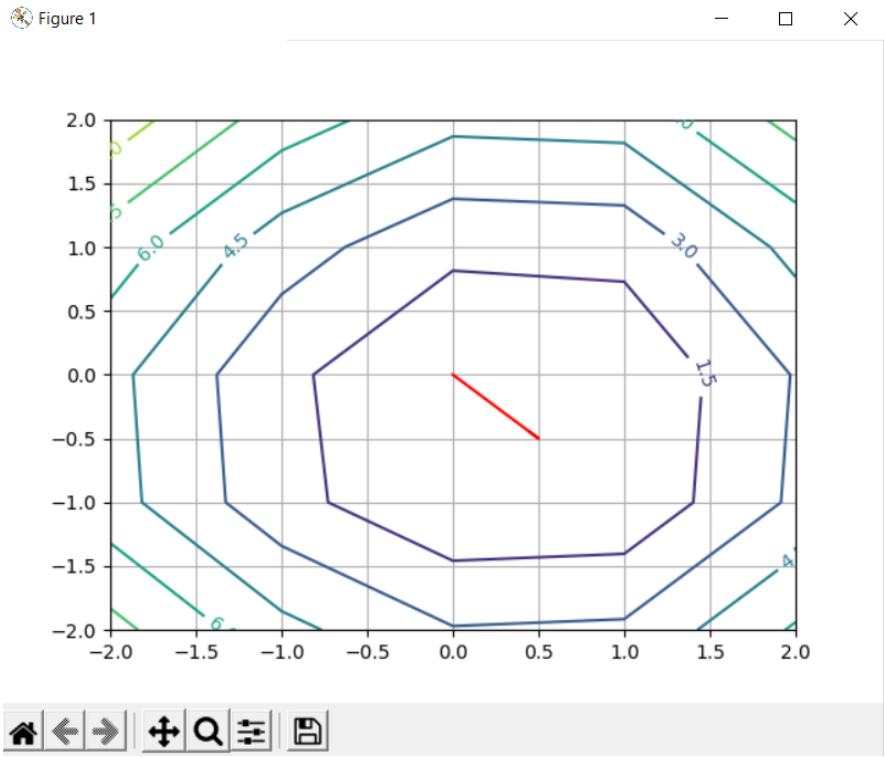
Step	x	y
1	1,666666667	-1,333333333
	Step count = 1	

Поточечная эмуляция:



Результаты для второй функции:

Step	x	y
1	0,5	-0,5
2	0,450626693	-0,450626693
3	0,450183648	-0,450183648
	Step count = 3	



Метод Ньютона обеспечивает очень высокую скорость сходимости для функций очень близких к квадратичным. Как видно из последнего графика, для множества других функций метод ведет себя нестабильно и медленно сходится. Также стоит отметить, что для этого метода главные условия, что Гессиан в текущей точке должен быть положительно определен и квадратичная аппроксимация должна быть адекватным приближением оптимизируемой функции.

Задание 4

Проще всего сгенерировать случайную квадратичную функцию размера n, взяв диагональную матрицу, так как диагональные матрицы эффективно работаю при больших значения n, в нашей матрице диагональные элементы сгенерированы случайно, и

находятся в пределах от 1 до числа обусловленности k, по условию k <= 1. По матрице восстанавливаем функцию.

Примеры матриц и функций:

```
0.936 0.000 0.000
0.000 0.780 0.000
0.000 0.000 0.869
 0.9360974448776489 * x[0] ** 2 + 0.7798288640433019 * x[1] ** 2 + 0.8690657344855709 * x[2] ** 2

Condition number: 1.2003883006126712
```

```
0.334 0.000 0.000 0.000 0.000
0.000 0.441 0.000 0.000 0.000
0.000 0.000 0.717 0.000 0.000
0.000 0.000 0.000 0.533 0.000
0.000 0.000 0.000 0.000 0.139
 0.3342548463565229 * x[0] ** 2 + 0.4405521335996706 * x[1] ** 2 + 0.7169735681241526 * x[2] ** 2 + 0.5331815571079843 * x[3] ** 2 + 0.1387930141499215 * x[4] **

Condition number: 5.165775615692672
```

```
0.883 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.768 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.118 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.773 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.548 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.942 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.967 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.973 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.489 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.039
 0.8831357372064648 * x[0] ** 2 + 0.7683562630633453 * x[1] ** 2 + 0.1175530953607492 * x[2] ** 2 + 0.7734619456821937 * x[3] ** 2 + 0.5483292481479474 * x[4] **

Condition number: 24.789795707554955
```

Пример применения метода и получения количества итераций:

Для n = 3 итераций: 70

```
0.944 0.000 0.000
0.000 0.814 0.000
0.000 0.000 0.584
 0.9442955761193699 * x[0] ** 2 + 0.8135788586707858 * x[1] ** 2 + 0.5839006844612815 * x[2] ** 2

Condition number: 1.6172195053865983

After optimisation:
Count of steps: 70
```

Для n = 5 итераций: 184

```
0.647 0.000 0.000 0.000 0.000
0.000 0.586 0.000 0.000 0.000
0.000 0.000 0.145 0.000 0.000
0.000 0.000 0.000 0.877 0.000
0.000 0.000 0.000 0.000 0.895
 0.6472402777975397 * x[0] ** 2 + 0.5862574676307069 * x[1] ** 2 + 0.14519944351754044 * x[2] ** 2 + 0.8768985175217872 * x[3] ** 2 + 0.8951031477800561 * x[4] *

Condition number: 6.164645856042316

After optimisation:
Count of steps: 184
```

PyCharm 2021.1.1 available

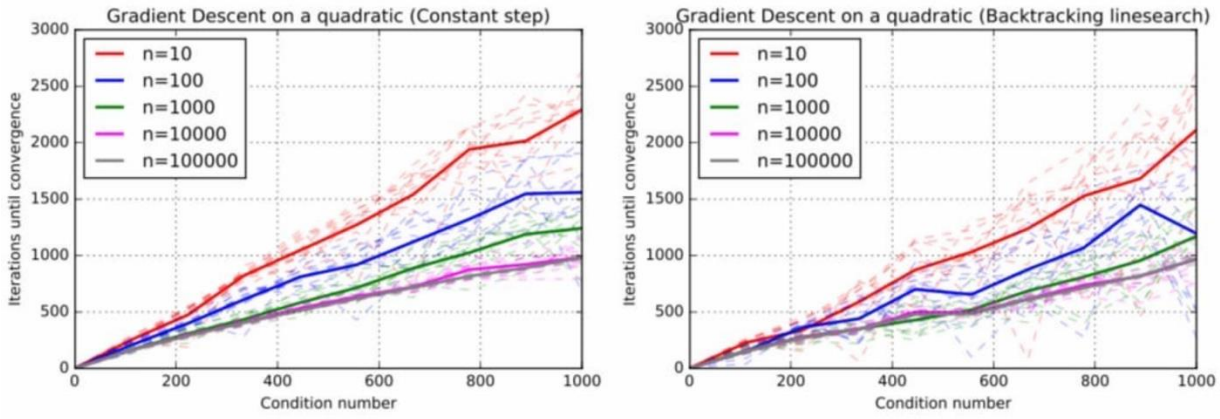
Для n = 10 итераций: 242

```
0.129 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.465 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.653 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.338 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.576 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.732 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.114 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.150 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.447 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.732
 0.12942919909209427 * x[0] ** 2 + 0.4651365203150116 * x[1] ** 2 + 0.6531411032549012 * x[2] ** 2 + 0.33804624953969786 * x[3] ** 2 + 0.5756037641583169 * x[4] *

Condition number: 6.420498818980573

After optimisation:
Count of steps: 242
```

То есть мы фиксируем n перебираем различные числа обусловленности k по сетке и строим график. Получаем семейство кривых зависимостей от n и k. Затем для других n и повторяем эти действия



Перебрав различные случаи, можно сделать следующие выводы: Зависимость числа итераций, необходимое градиентному спуску для сходимости от числа обусловленности k получилась линейной. С ростом размерности пространства n оптимизируемых переменных число итераций не увеличивается.

Вывод:

Проанализировав работу различных методов на наших функция, мы сделали выводы, что наиболее быстрым и массово применимым можно считать метод наискорейшего спуска. Он сочетает в себе лучшее от других методов и обладает свойством обязательной сходимости. Метод Ньютона также показал отличные результаты с квадратичными функциями, поэтому именно для них он наиболее целесообразен. Однако, он имеет жесткие требования к функция, поэтому лучше ради достижения сходимости за один шаг использовать квадратичную функцию, но на других не стоит. Также стоит заметить, что при оптимизации функций общего вида, метод сопряженных направлений сходится быстрее метода наискорейшего спуска.

Ссылка на лабу:

https://github.com/is-y23/applied-math-team-45