Федеральное государственное автономное образовательное учреждение высшего образования
Национальный исследовательский университет

"Высшая школа экономики"

Образовательная программа "Прикладная математика
бакалавр

ОТЧЕТ
по проектной работе

Метод сопряженных градиентов поиска минимума функции

Выполнила студентка гр. БПМ-214
Александра Даниловна Мухина

Руководитель проекта
Загвоздина Ксения Олеговна

10
(оценка)

(подпись)

Москва - 2022 г.

Содержание

1 Введение

Методы оптимизации позволяют решать множество задач поиска оптимального решения: нахождение оптимальной технологии, геометрической конструкции, времени технологических процессов и подобных задач в самых разных областях науки и техники. Часто в такого рода прикладных задачах встречается проблема нахождения именно минимума функции, к примеру, нахождение минимума ошибки, погрешности и тому подобное. По этой причине в данной работе рассматривается задача нахождения минимума, кроме того, к задаче нахождения максимума перейти несложно.

2 Постановка задачи

В данном проекте первой задачей станет изучение метода наискорейшего спуска для решения систем линейных уравнений. После будет рассмотрено применение изученного материала для выполнения основной задачи проекта - поиска минимума функции квадратичного вида. Также для углубления в методы оптимизации и изучения более эффективных методов в работу будут включены изучение теории, реализация и проверка на практике работы метода сопряженных градиентов. Для того, чтобы удостовериться в эффективности данного метода необходимо будет сравнить его с методом наискорейшего спуска.

Также будет изучен метод Полака-Рибьера, который является обобщенным методом сопряженных градиентов, подходящим для неквадратичных функций. Однако, работа концентрируется на самих методах, ввиду этого алгоритмы будут реализованы для квадратичных функций, взятых как упрощенный вариант, от которого несложно перейти к более сложному. Подобных усовершенствований метода сопряженных градиентов множество и моя коллегой изучит метод Ньютона-Рафсона. Поэтому, взяв одни данные и обменявшись результатами работы, нами произведется сравнение результатов работы изученного мною алгоритма Полака-Рибьера с методом Ньютона-Рафсона.

Все изученные методы будут реализованы на языке Python без использования каких-либо встроенных функций, заменяющих алгоритмы оптимизации, с использованием следующих библиотек: matplotlib.pyplot, numpy, time.

3 Метод сопряженных градиентов поиска минимума функции

3.1 Наискорейший спуск

Задача решения системы линейных уравнений $Ax = b$ также решается методами оптимизации, рассмотрим метод наискорейшего спуска. Это алгоритм нахождения максимально приближенного решения системы уравнений. Известны значения матрицы коэффициентов A, столбец свободных членов b и столбец стартовых(предполагаемых) решений x. Приближение значений x происходит с некоторым шагом: $x = x + \alpha * r$, где $r = b - Ax$ - невязка. Очевидно, что результат и скорость приближения зависят от выбираемого шага α, который можно задать постоянным вручную, но практичнее высчитывать заново на каждом шаге по данной формуле:

$$\alpha_{(i)} = \frac{r_{(i)}^T r_{(i)}}{r_{(i)}^T A r_{(i)}}$$

Чтобы оценить приближенность решений, рассчитывается значение нормы невязки, ошибки по формуле $\delta = r^T r$, которая представляет из себя длину вектора невязки. Таким образом, на каждом шагу вычисляется значение невязки, шага α, пересчитываются значения x до тех пор пока значение ошибки не станет допустимо малым.

Чтобы перейти от задачи решения системы линейных уравнений к задаче поиска минимума функции, рассмотрим понятия квадртичной функции и градиента. Квадратичная функция имеет вид

$$f(x) = \frac{1}{2} x^T A x - b^T x + c$$

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \quad b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \quad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

причем в нашей работе матрица A - симметричная и положительно определенная.

Градиент - это вектор наискорейшего возрастания функции, направлением указывающий в сторону роста значений функции, а по величине равный быстроте роста функции в этом направлении. Рассчитывается градиент из частных производных:

$$\operatorname{grad} u = \left(\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y} \right)$$

Рассмотрим изучаемую функцию. Раскладывая ее следующим образом

$$f(x) = \frac{1}{2} (x_1 x_2) \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - (b_1 b_2) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + c =$$

$$= \frac{1}{2}\left(x_1 \cdot a_{11} + x_2 a_{21} \quad x_1 a_{12} + x_2 a_{22}\right)\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - b_1 x - b_2 x_2 + c =$$

$$= \frac{1}{2}a_{11}x_1^2 + \frac{1}{2}\left(a_{21} + a_{12}\right)x_1 x_2 + \frac{1}{2}a_{22}x_2^2 - b_1 x_1 - b_2 x_2 + c$$

Найдем градиент

$$\text{gradf} = \left(\frac{\partial f}{\partial x_1}; \frac{\partial f}{\partial x_2}\right) =$$

$$= \left(a_{11}x_1 + \frac{1}{2}\left(a_{21} + a_{12}\right)x_2 - b \quad \frac{1}{2}\left(a_{21} + a_{12}\right)x_1 + a_{22}x_2 - b_2\right)$$

С учетом симметричности матрицы A заметим, что градиент функции равен

$$\text{gradf} = Ax - b$$

Перед нами стоит задача поиска минимума подобной функции. Исходя из того, что отсутствие изменения значений функции означает точку экстремума, данную проблему можно свести к решению уравнения $\text{grad}\, f = 0$. Таким образом наша задача сведена к решению системы линейных уравнений $Ax = b$.

Зная всё выше сказанное, разберемся в теории нахождения шага α. Вектор r, который мы раннее назвали невязкой, является антиградиентом квадратичной функции, то есть вектором, противоположно направленным градиенту. Обратимся к графику линий уровня функции (Рис.1). Градиент - это перпендикуляр к касательной в точке значений x_0 к линии уровня, направленный от центра. В алгоритме мы двигаемся по прямой в сторону антиградиента, указанного на рисунке стрелкой. И выбор α - выбор насколько продвинуться вперед по этой прямой до следующего значения x_1.

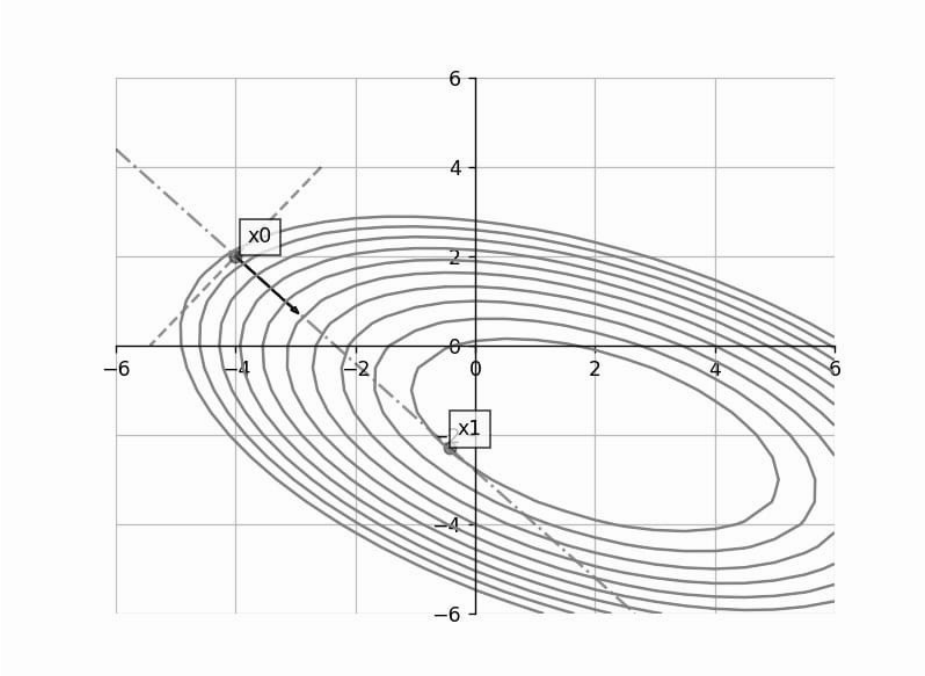

Рис. 1: Градиент и линии уровня квадратичной функции

Оказывается, наилучшим выбором будет перейти в точку минимума параболы, расположенной вдоль этой прямой, решив задачу одномерной оптимизации. Запишем функцию $g(\alpha) = f(x_0 + \alpha r_0)$, распишем ее, сгруппируем и решим задачу нахождения минимума этой функции, приравняв производную к нулю. Получим значение шага

$$g(\alpha) = f\left(x_0 + \alpha r_0\right) = \frac{1}{2}\left(r_0 + \alpha r_0\right)^\top A\left(x_0 + \alpha r_0\right) - b^\top\left(x_0 + \alpha r_0\right) + c$$
$$= \frac{1}{2}\alpha^2 r_0^\top A r_0 + r_0^\top\left(Ax_0 - b\right)\alpha + \left(\frac{1}{2}x_0^\top A x_0 + x_0^\top r_0 + c\right)$$

$$g'\left(\alpha\right) = \left(d_0^\top A d_0\right)\alpha + d_0^\top\left(Ax_0 - b\right) = 0$$
$$\alpha = -\frac{d_0^\top\left(Ax_0 - b\right)}{d_0^\top A d_0} = \frac{d_0^\top d_0}{d_0^\top A d_0}.$$

После в алгоритме всё повторяется заново, высчитывается новый градиент, перпендикулярный прошлому, новый шаг α и новое приближение x.

$$r_{(i)} = b - Ax_{(i)},$$
$$\alpha_{(i)} = \frac{r_{(i)}^T r_{(i)}}{r_{(i)}^T A r_{(i)}},$$
$$x_{(i+1)} = x_{(i)} + \alpha_{(i)} r_{(i)}.$$

Таким образом некой "лесенкой"алгоритм приведет нас к центру эллипсов, то есть к минимуму изображенной функции (Рис. 2).

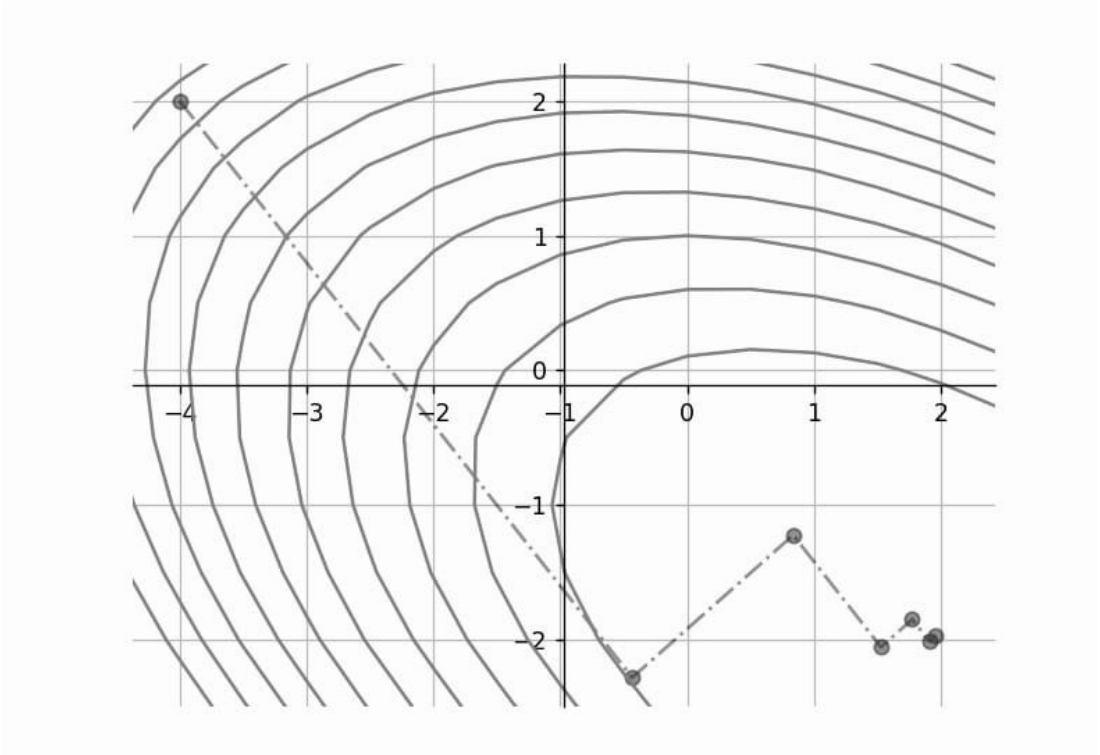

Рис. 2: Метод наискорейшего спуска

Можно предположить, что шаг по перпендикуляру не является эффективным и нуждается в большом количестве приближений, чтобы достигнуть удовлетворительного результата.

3.2 Метод сопряженных градиентов

Было бы практичнее, если бы учитывалось прошлое направление приближения и с учетом его строилось новое. Именно на этом основывается принцип работы метода сопряженных градиентов.

В данном методе мы так же имеем начальные значения x_0, матрицу A и столбец b. Так же как в прошлом методе выбираем первое направление $d_0 = b - Ax_0$ - антиградиент и считаем α по прежним соображениям и формуле:

$$\alpha_0 = \frac{d_0^\top d_0}{d_0^\top A d_0}.$$

Далее произведем первое приближение $x_1 = x_0 + \alpha_0 r_0$.

Теперь рассмотрим понятие сопряженных векторов. Положительно определенная матрица позволяет ввести норму вектора следующим образом:

$$\|x\|^2 = (x, Ax) > 0 \text{ при } \boldsymbol{x} \neq 0.$$

Определение означает, что под скалярным произведением двух векторов $\boldsymbol{x}$ и y теперь подразумевается величина $(\boldsymbol{x}, A\boldsymbol{y})$. Векторы, ортогональные в смысле этого скалярного произведения

$$(x, Ay) = 0,$$

называют сопряженными (по отношению к данной матрице A).

Таким образом мы можем ввести понятие ортогональности векторов в нашем случае, относительно матрицы A: $x^\top Ay = 0$ и делать "ортогональные"шаги. Наша задача в идеале прийти к истинному значению x, начиная из $_0$ за два шага, которые будут A-ортогональны между собой. Заметим, что фактически это разложение в базис, просто выбранный определенным образом. Следовательно наша задача разложить вектор от начальной точки в конечную в базис A-ортогональных векторов.

Вернемся к выбору следующего направления приближения d_1. Для его нахождения будем использовать антиградиент в текущей точке и предыдщее направление d_0 с некоторым коэффициентом β.

$$d_1 = -\nabla f(x_1) + \beta_0 d_0$$

Из вышесказанного следует, что этот коэффициент следует подбирать таким образом, чтобы d1 и d0 были A-ортогональны. Запишем определение A-ортогональности для нашей задачи:

$$d_1^\top A d_0 = 0$$

Подставим d_1 и найдем β

$$d_1^\top A d_0 = -\nabla f\left(x_1\right)^\top A d_0 + \beta_0 d_0^\top A d_0 = 0$$

$$\beta_0 = \frac{\nabla f\left(x_1\right)^\top A d_0}{d_0^\top A d_0}$$

Далее опять используя одномерную минимизацию находим следующее приближение x_i, направление d_i A-ортогональное d_{i-1} и так далее до тех пор, пока значение ошибки $\delta = r^\top r$ не станет допустимо малым. Получаем результат всего за n шагов.

3.3 Метод сопряженных градиентов Полака-Рибьера

Метод Полака-Рибьера - это обобщение метода сопряженных градиентов для неквадратичных функций. Для совершения такого перехода рассмотрим градиенты квадратичных функций.

$$\boldsymbol{x}_{k+1} - \boldsymbol{x}_k = \boldsymbol{c}\boldsymbol{d}_k$$

$$\nabla f\left(x_{k+1}\right) - \nabla f\left(x_k\right) = \left(Ax_{k+1} - b\right) - \left(Ax_k - b\right) = A\left(x_{k+1} - x_k\right) = cAd_k$$

А теперь заменим:

$$Ad_k = \frac{1}{c}\left(\nabla f\left(x_{k+1}\right) - \nabla f\left(x_k\right)\right)$$

Таким образом, мы избавились от специфичного вида квадратичной функции и можем перейти к использованию обычных градиентов любой функции. На этом и строится метод Полака-Райбера. Существует ряд вариаций метода сопряженных градиентов для неквадратичных функций, но конкретно в этом производится замена β на выражение из градиентов, подходящее функции любого вида

$$\beta_k = \frac{\nabla f\left(x_{k+1}\right)^\top \left(\nabla f\left(x_{k+1}\right) - \nabla f\left(x_k\right)\right)}{d_k^\top \left(\nabla f\left(x_{k+1}\right) - \nabla f\left(x_k\right)\right)}$$

Полученная формула называется формулой Полака-Рибьера. Причем необходимо, чтобы параметр β всегда оставался положительным, поэтому алгоритм начинается заново если параметр отрицателен. Также в методе используется матрица Гессе - матрица вторых производных функции.

$$f''(\bar{x}) = \begin{pmatrix} \dfrac{\partial^2 y}{\partial x_1 \partial x_1} & \dfrac{\partial^2 y}{\partial x_1 \partial x_1} \\ \dfrac{\partial^2 y}{\partial x_1 \partial x_1} & \dfrac{\partial^3 y}{\partial x_1 \partial x_1} \end{pmatrix}$$

Направление приближения высчитывается не только с помощью градиента, но и с обратным гауссианом следующим образом: $d = M^{-1}r$ А значение ошибки $\delta = r^T d$

Алгоритм так же завершается когда максимальное количество итераций, задаваемое вручную превышается, либо когда значение ошибки удовлетворительно мало.

Для применения данного метода к квадратичным функциям стоит вспомнить лишь, что градиент функции данного вида равен $Ax - b$ и посчитать матрицу Гесса, расписав функцию:

$$A = \begin{pmatrix} a_{11} & a \\ a & a_{22} \end{pmatrix} \quad f(x) = \frac{1}{2}x^\top Ax - b^\top x + c$$

$$f''(x) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} \end{pmatrix} = \begin{pmatrix} a_{11} & \frac{1}{2}(a+a) \\ \frac{1}{2}(a+a) & a_{22} \end{pmatrix} = A$$

Следовательно, A является матрицей Гессе.

4 Результаты численных экспериментов

4.1 Наискорейший спуск

Для изучения удачного выбора постоянного шага *alpha* и сравнения с алгоритмом с изменяемым шагом были построены графики зависимости нормы невязки d от выбора α для трех матриц разных размерностей: 2x2, 3x3, 4x4. Рассмотрим матрицу 2x2 $A = \begin{pmatrix} 2 & 3 \\ 3 & 9 \end{pmatrix}$, вектор $b = \begin{pmatrix} 6 \\ 14 \end{pmatrix}$, истинные значения стобца $x = \begin{pmatrix} 1.33 \\ 1.11 \end{pmatrix}$, начальные значения столбца $x = \begin{pmatrix} -2 \\ 3 \end{pmatrix}$

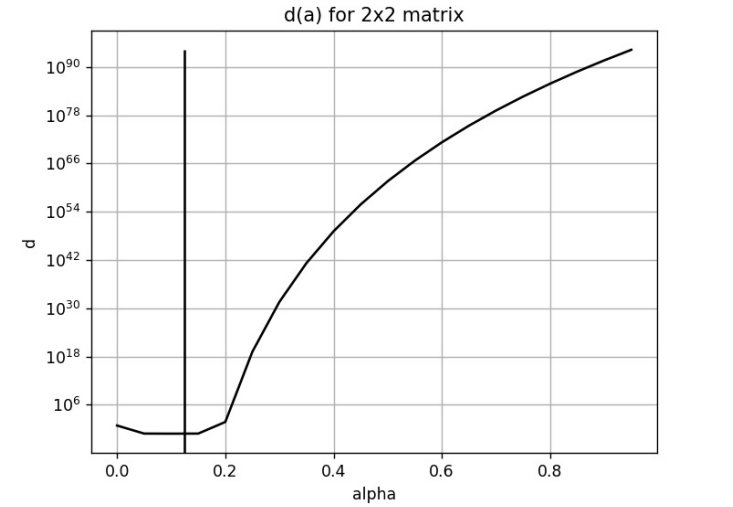

Рис. 3: График зависимости значения ошибки от α для матрицы 2x2

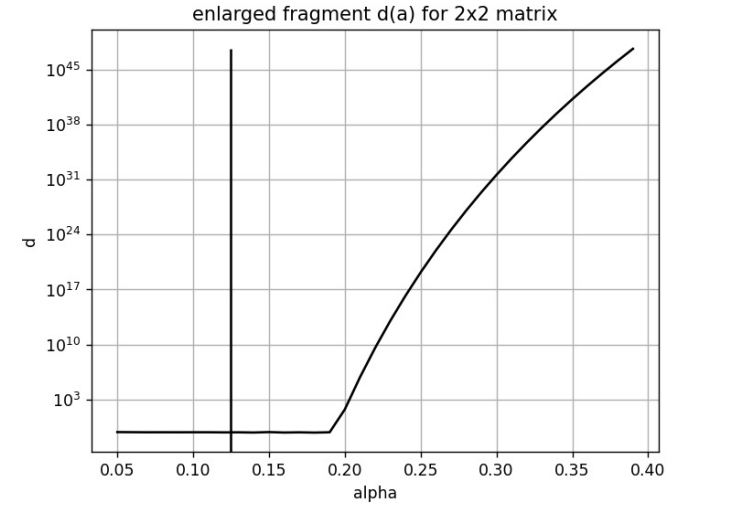

Рис. 4: Увеличенный фрагмен Рис. 3

Заметим, что начиная с некоторого α происходит резкое увеличение погрешности. Это обсуловлено тем, что выбирая данный шаг или больше, алгоритм "перескакивает"искомые значения и с каждым новым шагом удаляется, а не приближается к ним.

Вертикальной прямой обозначено первое значение альфа из алгоритма с переменным шагом. Шаг высчитывается разумно и даёт результат сразу же с достаточно хорошей точностью. Однако стоит заметить, что его можно увеличить без потери точности, что отчетливо видно на увеличенном фрагменте (Рис. 4).

Обратимся к численным результатам. Метод с постоянным шагом $\alpha = 0.13$, равным высчитываемому на первом шаге в алгоритме для переменного шага дал результат с нормой невязки 0.066 за 32 итерации. Метод в изменяемым шагом дал результат с нормой невязки 0.0484 за 19 итераций. Допустимая погрешность была равна 0.01. Норма невязки алгоритма с изменяемым шагом меньше, причем за меньшее количество итераций, что доказывает превосходство данного метода.

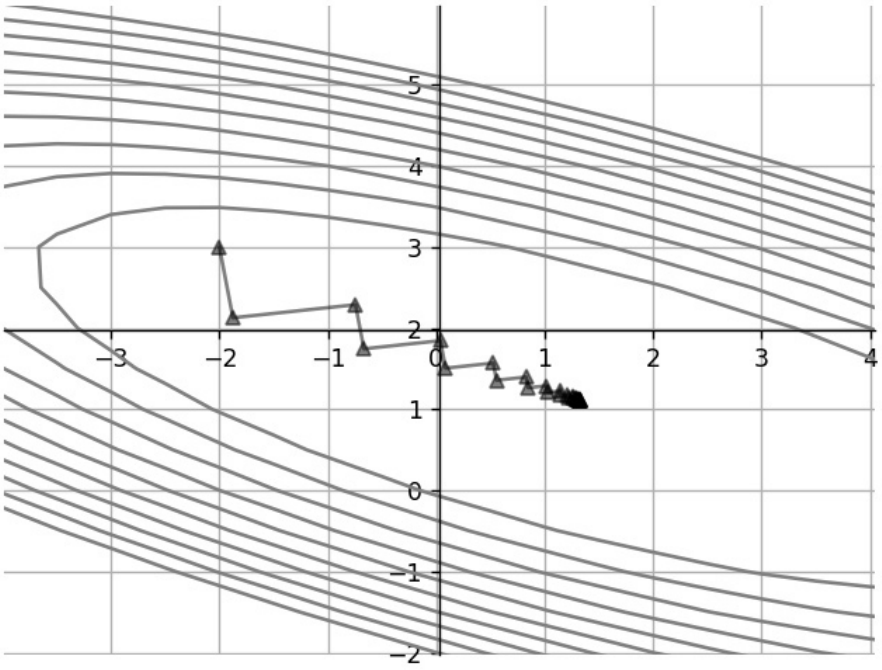

Рис. 5: Траектория приближения наискорейшим спуском

Также на Рис. 5 можем увидеть ту самую "лесенку"наискорейшего спуска с изменяемым шагом, описанную в теоретической части.

4.2 Метод сопряженных градиентов

Для просмотра результатов работы метода сопряженных градиентов возьмем те же данные, что и для наискорейшего спуска. Алгоритм дал результат с нормой невязки неотличимой от машинного нуля за 2 итерации при допустимой погрешности 0.01. Очень хороший результат, метод позволяет найти значения x с высокой точностью за столь малое количество итераций, что конечно дает ему преимущество перед методом наискорейшего спуска.

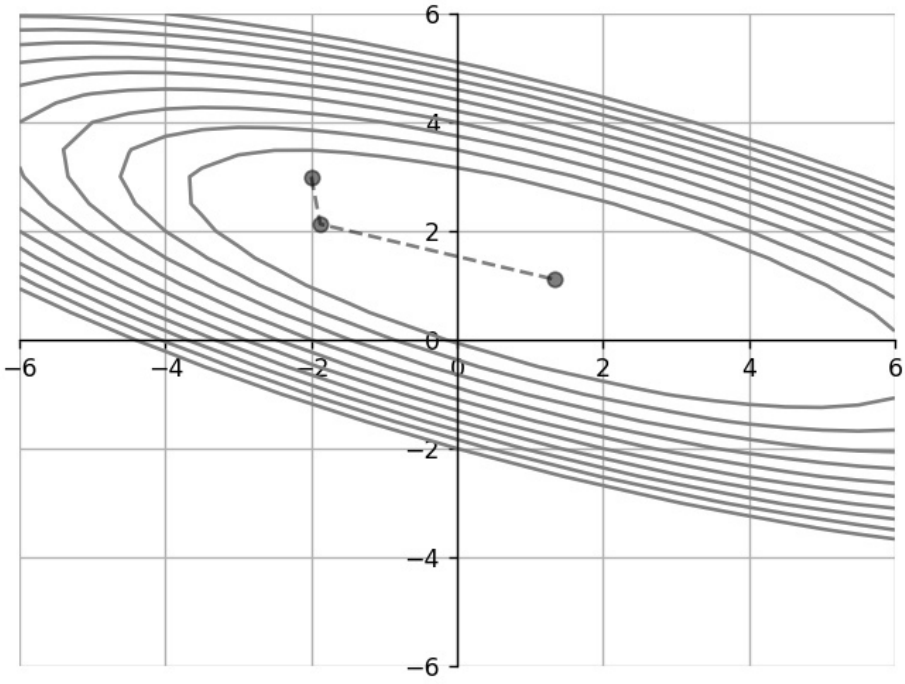

Рис. 6: Траектория приближения методом совместных градиентов

Шаги алгоритма можно увидеть на графике представленном на Рис.6 и действительно всего за 2 шага в A-сопряженных направлениях траектория заканчивается в центре линий уровня, то есть в минимуме функции.

4.3 Метод сопряженных градиентов Полака-Рибьера

На тех же данных рассмотрим работу метода Полака-Рибьера. Достигнут результат с нормой невязки неотличимой от машинного нуля за 1 итерацию. Для изучения скорости работы алгоритма не в итерациях, а в фактическом времени, я также измерила время работы функций. И результат был получен за 0.0009973 секунд. С той же задачей алгоритм наискорейшего спуска дошел до ответа за 0.0010292 секунд и 29 итераций. Траектории приближения можем видеть на Рис.7, где пунктирной линией обозначен метод Полака-Рибьера, а сплошной - наискорейший спуск.

Для проверки мощности метода Полака-Рибьера испытаем его работу на матрице размерности 7. Алгоритм достиг результата точности 10^{-15} за 0.000998 секунд и 1 итерацию, в то время как метод наискорейшего спуска не достиг правильного результата за 1000 итераций и 0.01300 секунд.

Из чего мы можем сделать вывод, что метод Полака-Рибьера эффективнее метода наискорейшего спуска и справляется даже лучше, чем оригинальный метод сопряженных градиентов.

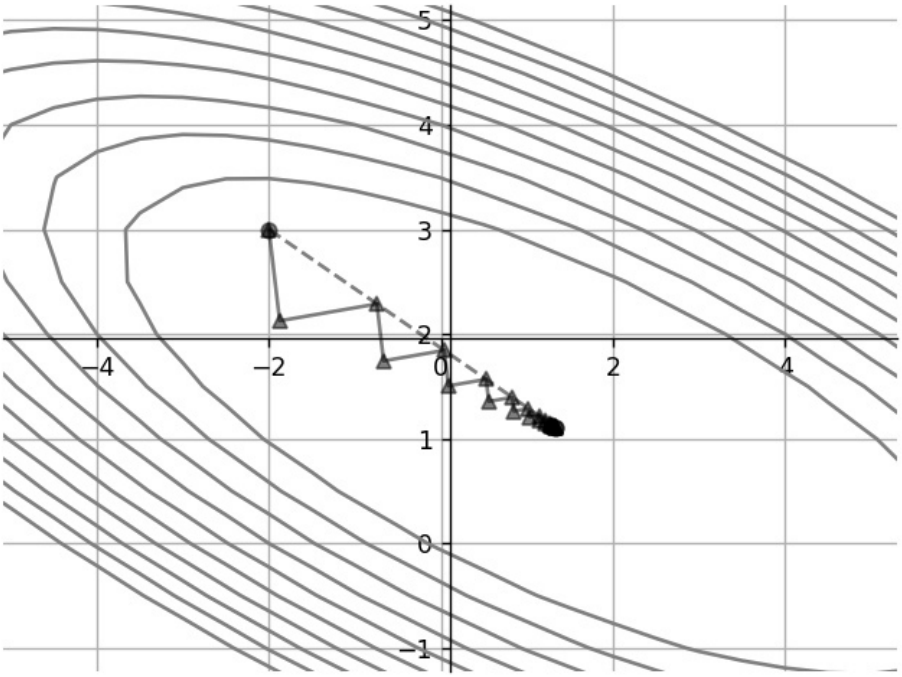

Рис. 7: Траектория приближения методом Полака-Рибьера и наискорейшего спуска

Теперь сравним методы Полака-Рибьера и Ньютона-Рафсона. С коллегой мы взяли одни данные и обменялись результатами. Алгоритм Полака-Рибьера достиг истинных значений за 1 итерацию с точностью 10^{-15}. Алгоритм Ньютона-Рафсона также достиг истинных значений за 2 итерации с точностью 10^{-14}. Благодаря пересчитыванию градиента и матрицы Гессе несколько раз и большей сложности вычислений, алгоритм Полака-Рибьера является более эффективным, но не сильно уступает методу Ньютона-Рафсона.

5 Выводы

В ходе работы была написана программа на языке Python и реализованы метод наискорейшего спуска с постоянным и изменяемым шагом α, метод сопряженных градиентов в общем виде и метод Полака-Райбера для квадратичных функций. Было проведено сравнение метода Полака-Рибьера с наискорейшим спуском, а также с методом Ньютона-Рафсона.

Было подтверждено на практике, что метод наискорейшего спуска с изменяемым шагом более эффективен, а вид траектории выведен согласно теории.

Метод сопряженных градиентов дал хороший результат с высокой точностью за гораздо меньшее количество шагов, чем подтвердил теоретическую выкладку.

Метод Полака-Рибьера имеет самую выскую эффективность и точность из всех выбранных методов методов, а также имеет возможность высичлять значение за одну итерацию. И даже является чуть более эффективным, чем метод Ньютона-Рафсона за счет большей сложности вычислений на каждом шагу.

6 Список литературы

Список литературы

[1] Jonathan Richard Shewchuk. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain :Edition 1 $\frac{1}{4}$ / J.R.Shewchuk - School of Computer Science Carnegie Mellon University, Pittsburgh, PA, 1994.-58c.-URL: https://math.nyu.edu/ greengar/painless-conjugate-gradient.pdf

[2] Python 3 documentation [Электронный ресурс].- URL: https://docs.python.org/3/

[3] Abbas Y. Al Bayati Iraqi Journal of Statistical Science / Abbas Y. Al Bayati Khalil K. Abbo Ibrahem A. Saleh. - The Fourth Scientific Conference of the College of Computer Science Mathematicspp, 2011. - pp[164-173]

[4] Н. Н. Калиткин Численные методы / под ред. А.А. Самарского. - М: "Наука 1978. - 508 с.

7 Приложение

Ссылка на репозиторий GitHub https://github.com/awwpurple/gradients