Национальный исследовательский университет ИТМО
Факультет информационных технологий и программирования
Прикладная математика и информатика

Методы оптимизации

Отчёт по лабораторной работе №3

Выполнили:
Ночевкина Наталья, M3233
Чуракова Александра, M3232
Ильченко Артём, M3233

Санкт-Петербург
2023

Gauss-Newton method

Метод Гаусса-Ньютона - итерационный численный метод поиска решения задачи наименьших квадратов, являющийся разновидностью метода Ньютона, решающего задачу минимизации $f(x) : \mathbb{R}^n \to \mathbb{R}$, т.е. нахождения такого $x_{min} = (x_0, \ldots, x_{n-1}) : \nabla f(x_{min}) = 0$.

В методе Ньютона используется формула обновления текущего значения аргумента x: $x_{n+1} = x_n - H^{-1}(x_n) \cdot \nabla f(x_n)$, где $H^{-1}(x_n)$ - обратный гессиан функции f

При решении задачи наименьших квадратов функция f представляется в виде:

$$f(x) = \frac{1}{2}||F(x)||^2 = \frac{1}{2}\sum_{i=0}^{n-1} F_i^2(x)$$

Тогда $\nabla f(x) = J^T(x) \cdot F(x)$, $J^T(x)$ - Якобиан f; $H(x) = J^T(x) \cdot J(x) + Q(x)$, где $Q(x) = \sum_{i=0}^{n-1} H_i(x) \cdot F_i(x)$, $H_i(x)$ - гессиан $F_i(x)$.

Исходя из вышеприведенных формул, получаем:

$$x_{n+1} = x_n - H^{-1}(x_n)\nabla f(x_n) =$$
$$= x_n - [J^T(x_n) \cdot J(x_n) + \sum_{i=0}^{n-1} H_i(x)F_i(x)]^{-1} \cdot J^T(x_n)F(x_n)$$

Метод Гаусса-Ньютона базируется на предположении, что $||J^T(x) \cdot J(x)|| \gg ||Q(x)||$: слагаемое $Q(x)$ принебрежимо мало в сравнение с $J^T(x) \cdot J(x)$, и потому им можно пренебречь. Тогда для метода Гаусса-Ньютона формула обновления аргумента будет иметь вид:

$$x_{n+1} = x_n - (J^T(x_n) \cdot J(x_n))^{-1} \cdot J^T(x_n)F(x_n)$$

Примеры работы

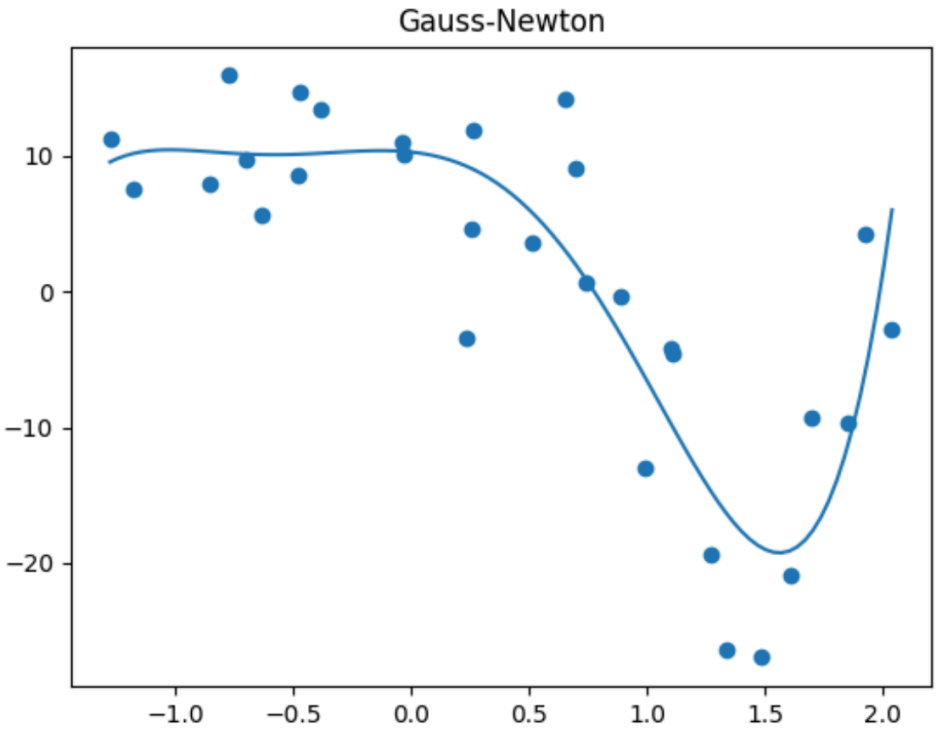


рис.1 восстановление $3x^5 + 0.8x^4 - 10x^3 - 8x^2 - x + 9$

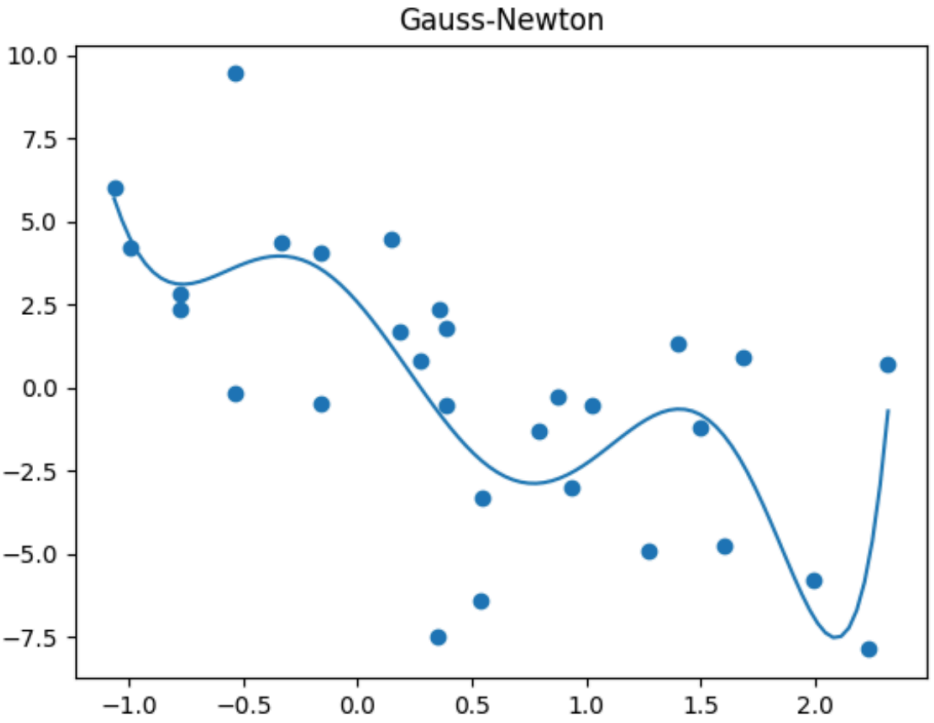


рис.2 восстановление $e^{x\sin^2(x)} - 4x$

Powell Dog Leg method

Powell Dog Leg - алгоритм, являющийся композицией метода Гаусса-Ньютона и Trust Region метода.

Trust Region method:

Trust region method - численный метод оптимизации, вычисляющий на каждой итерации шаг с помощью аппроксимации целевой функции ее квадратичной моделью, ограничивая область поиска оптимального решения величиной Δ_k, называемой ***trust region radius***

Таким образом, на каждой итерации метода вычисляем шаг (направление+величина), решая квадратичную задачу

$$\min_{p\in\mathbb{R}^2} m_k(p) = f(x_k) + p^T\nabla^2 f(x_k) + \frac{1}{2}p^T H(x_k)p,\ |p| \le \Delta_k$$

where $H(x_k) = \nabla^2 f(x_k)$ - hessian of the f

В методе Powell Dog Leg также перед началом работы алгоритма выбирается значение trust region radius Δ , задающее “доверительный” регион в n-мерном пространстве - гиперсферу радиуса Δ. Алгоритм работает также, как и метод Гаусса-Ньютона, но вычисление значения сигма-параметра

$\delta = (J^T(x_n)\cdot J(x_n))^{-1}\cdot J^T(x_n)\cdot F(x_n)$,

используемого для вычисления обновленного значения аргумента $x_{n+1} = x_n - \delta$, происходит с учетом ограничения на область поиска шага изменения - величину параметра Δ.

Если модуль вычисленного вектора δ не превосходит установленного значения Δ , то оставляем параметр δ без изменений, сразу вычисляя x_{n+1} . В противном случае вычислим вектор-параметр

$\delta_{sd} = J^T(x_n)\cdot f(x_n)$,

используемый в качестве изменения в алгоритме наискорейшего спуска, а также число

$$t = \frac{||\delta_{sd}||^2}{||J(x_n)\cdot\delta_{sd}||^2}$$

Если при $||\delta|| > \Delta$ выполнено $t\cdot||\delta_{sd}|| > \Delta$, то

$$\delta = \Delta\frac{\delta_{sd}}{||\delta_{sd}||}$$,

иначе $\delta = t\cdot\delta_{sd} + s\cdot(\delta' - t\cdot\delta_{sd})$, где $s : ||\delta|| = \Delta$, δ' принимает изначально вычисленное значение δ. Параметр s находится как корень уравнения квадратного уравнения вида

$ax^2 + bx + c = 0$, где

$a = ||\delta' - t\cdot\delta_{sd}||^2, b = 2\cdot(t\cdot\delta_{sd}, \delta' - t\cdot\delta_{sd}), c = \Delta^2 - ||t\cdot\delta_{sd}||^2$

$s = \frac{-b+\sqrt{D}}{2a}$ - , корень берется со знаком ‘+’.

Таким образом, значение шага изменения аргумента δ вычисляется следующим образом

$$\delta = \begin{cases} \delta, & ||\delta|| \leq \Delta \\ \Delta \cdot \frac{\delta_{sd}}{||\delta_{sd}||}, & ||\delta|| > \Delta, t \cdot ||\delta_{sd}|| > \Delta \\ t \cdot \delta_{sd} + s \cdot (\delta' - t\delta_{sd}), & otherwise \end{cases}$$

Примеры работы

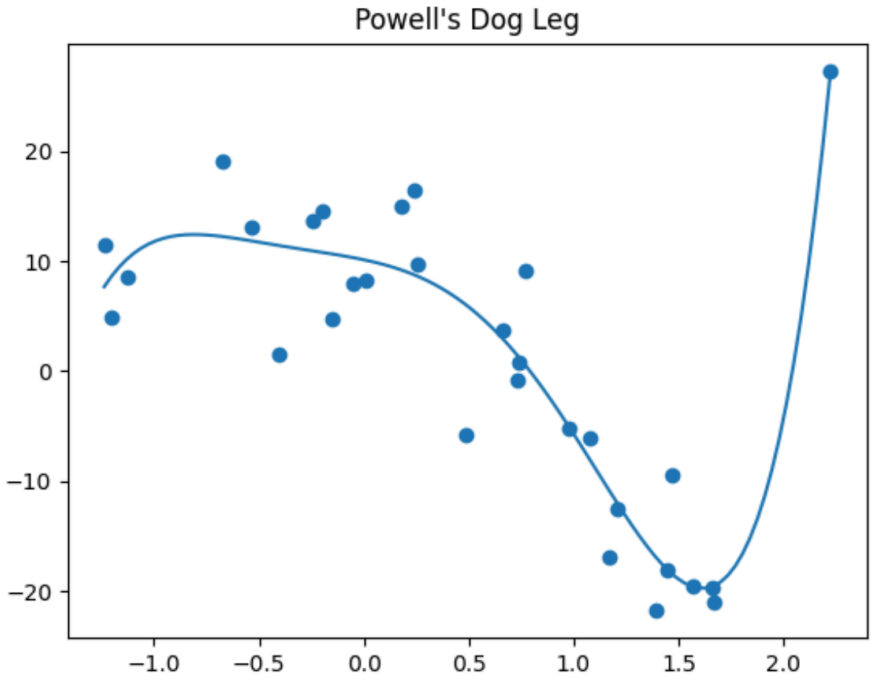


рис.3 восстановление $3x^5 + 0.8x^4 - 10x^3 - 8x^2 - x + 9$

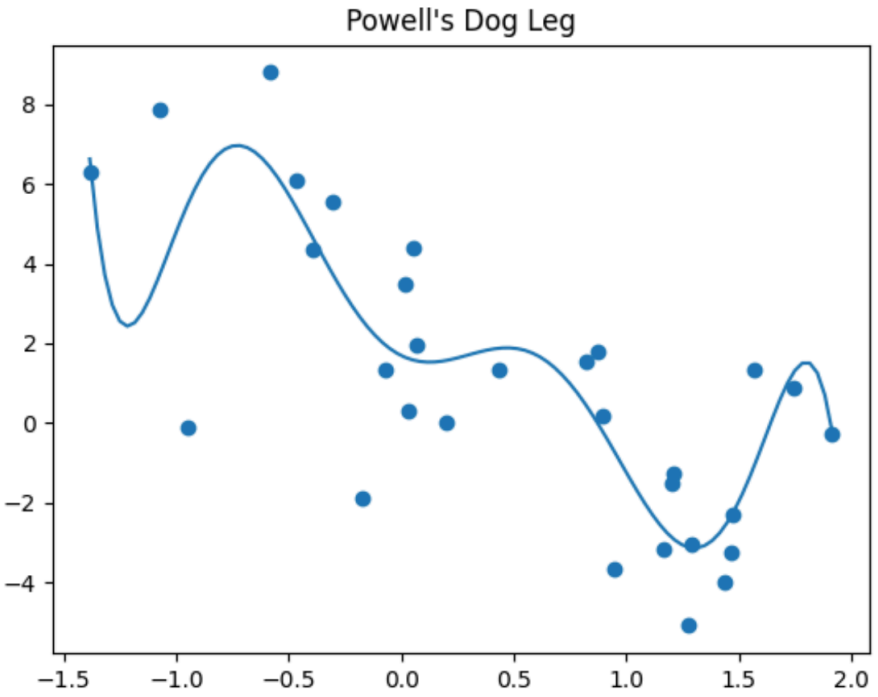


рис.4 восстановление $e^{x\sin^2(x)} - 4x$

Сравнение Powell Dog Leg и Gauss-Newton с полиномиальными регрессиями из прошлой лабораторной работы

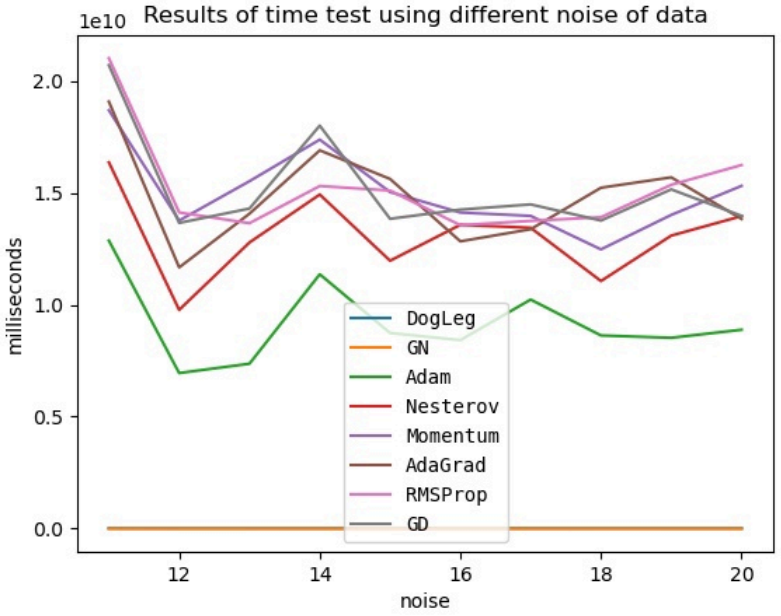
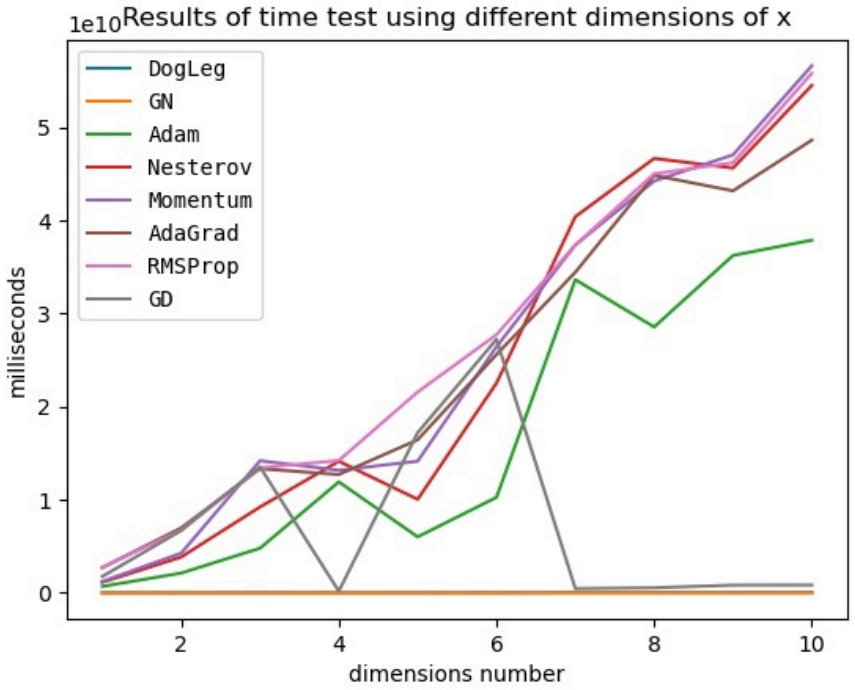
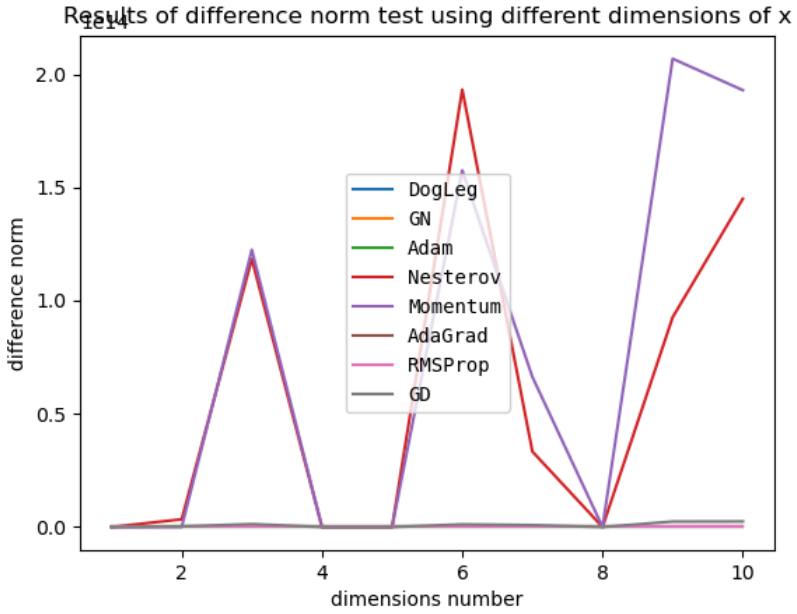


рис.5 сравнение различных методов по времени работы

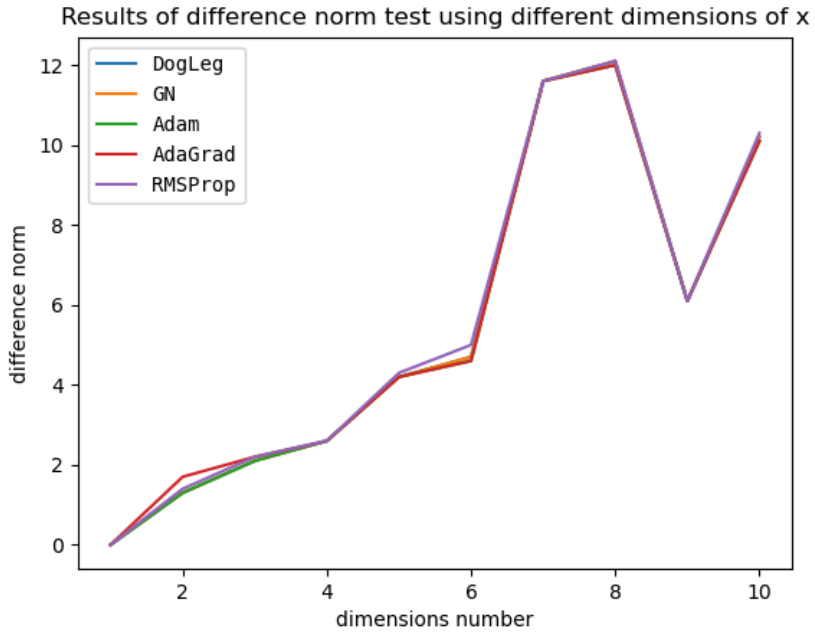
Из данного графика мы можем наблюдать, что метод Гаусса-Ньютона и Dog Leg отработали лучше остальных методов.



Интересно, что градиентный спуск в последние несколько измерений работает лучше остальных методов (кроме метода Гаусса-Ньютона) по времени.

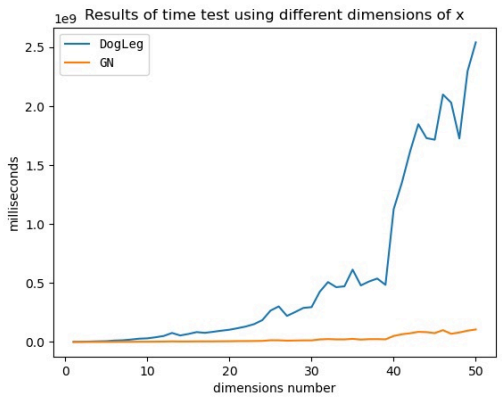


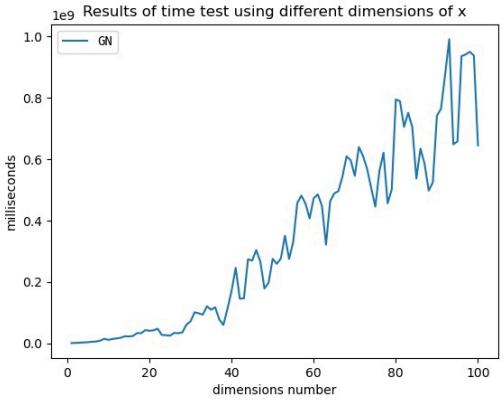
Результаты некоторых методов вышли за пределы допустимых значений (т. е. устремились к бесконечности), однако Dog Leg и метод Гаусса-Ньютона отработали согласно ожиданиям, т. е. наша реализация оказалась устойчивой.



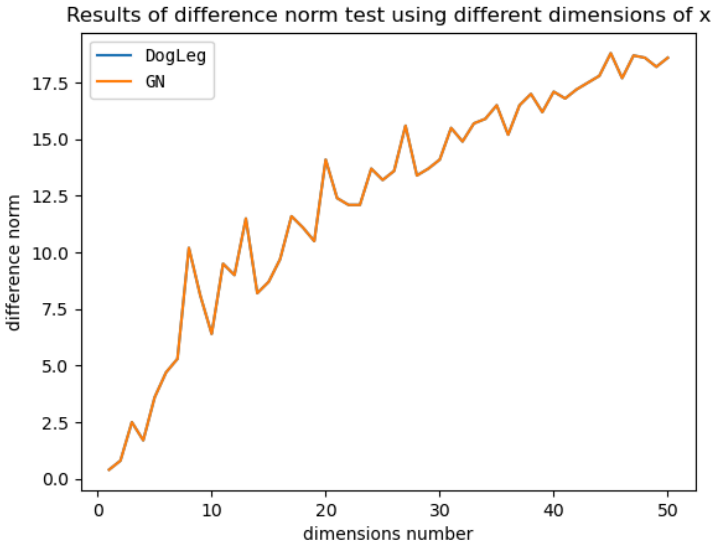
Без методов, которые не справились с задачей, остальные показали отличный и стабильный результат.

Тесты для методов Гаусса-Ньютона и Powell Dog Leg

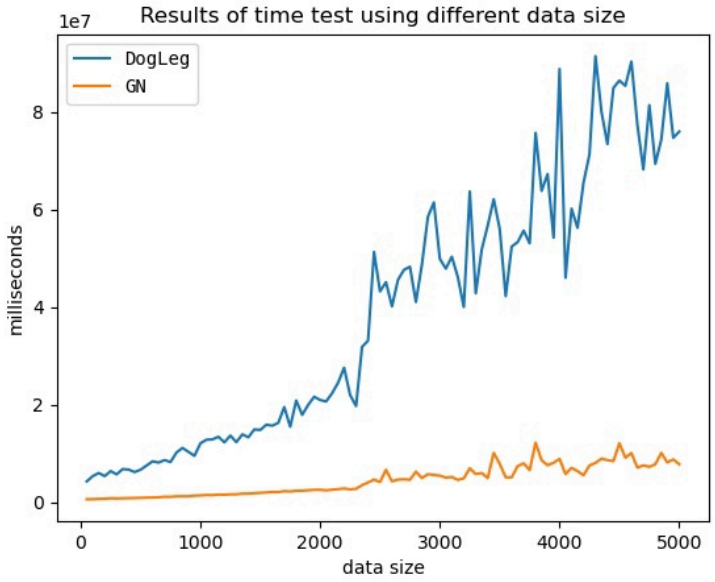




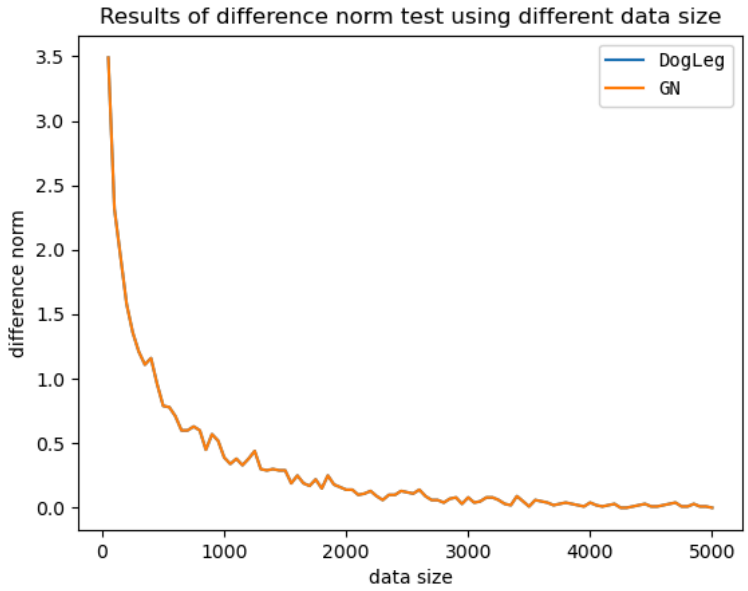
При большом количестве измерений Dog Leg начинает работать хуже, чем метод Гаусса-Ньютона. Для большей наглядности мы привели график работы метода Гаусса-Ньютона отдельно.



Здесь можно наблюдать полное совпадение графиков Dog Leg и Gauss-Newton. Это значит, что оба метода показали идеальный результат.



При больших объемах данных Dog Leg начинает вести себя хуже, чем метод Гаусса-Ньютона.



Снова полное совпадение графиков. Оба метода ведут себя просто идеально.

BFGS method

BFGS (Broyden-Fletcher-Goldfarb-Shanno algorithm) - итерационный метод численной оптимизации, относящийся к классу квазиньютоновских методов и решающий задачу поиска минимума нелинейного функционала.
Метод осуществляет поиск минимума, вычисляя на каждой своей итерации значение коэффициента α , удовлетворяющего Strong Wolfe Conditions:

- $f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k$
- $|\nabla f(x_k + \alpha_k p_k)^T p_k| \leq |c_2 \nabla f_k^T p_k|$

В качестве c_1 и c_2 были взяты значения 10^{-4} и $0,9$, соответственно. Подробнее с реализацией можно ознакомиться по ссылке.
После нахождения α текущее значение аргумента функции (вектора координат x_k) изменяется на $\alpha_k p_k$:

$x_{k+1} = x_k + \alpha_k p_k$, где p_k - точка, в направлении которой на следующей итерации будет осуществляться поиск; ее значение вычисляется как произведение приближенного обратного гессиана функции на оператор Набла, взятое со знаком минус.

$p_k = -H_k \cdot \nabla f_k$,

где H_k - приближенное значение обратного гессиана функции, вычисляемое следующим путем:

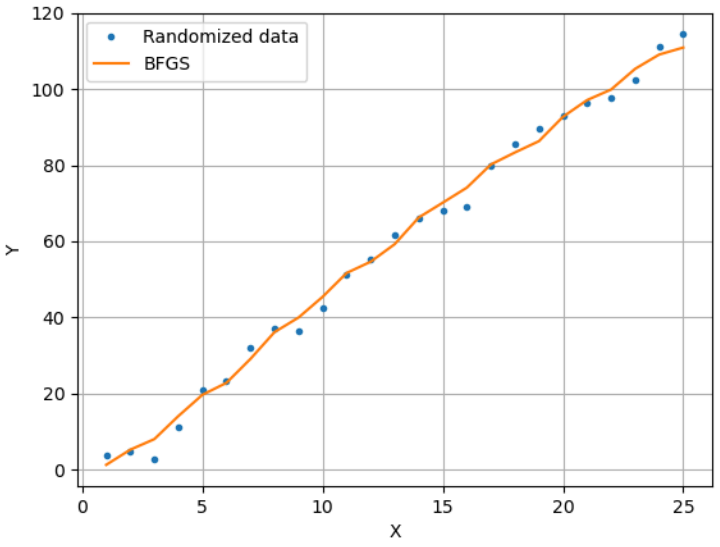
$H_k = l_k \cdot H_{k-1} \cdot r_k + ((\nabla f_k - \nabla f_{k-1})^T \cdot \alpha_k p_{k-1})^{-1} \cdot (\alpha_k p_{k-1} \cdot (\alpha_k p_{k-1})^T)$

$l_k = H_0 - ((\nabla f_k - \nabla f_{k-1})^T \cdot \alpha_k p_{k-1})^{-1} \cdot (\alpha_k p_{k-1} \cdot (\nabla f_k - \nabla f_{k-1})^T)$

$r_k = H_0 - ((\nabla f_k - \nabla f_{k-1})^T \cdot \alpha_k p_{k-1})^{-1} \cdot ((\nabla f_k - \nabla f_{k-1}) \cdot (\alpha_k p_{k-1})^T)$

За счет использования обратного Гессиана метод на каждом шаге также оценивает кривизну функции, что позволяет определять с большей точностью направление убывания функции, а вычисление коэффициента α с применением Strong Wolfe Conditions позволяет подобрать оптимальную величину шага, чтобы минимизировать возможные отклонения от траектории наискорейшего спуска.
Нахождение обратного гессиана не замедляет существенно работу метода, потому как значение матрицы Гессе вычисляется апромаусиционно, в силу чего не требует вычислений кубической сложности.
Применение Strong Wolfe Conditions, вместо обычных, обусловлено тем, что усиленные условия, в отличие от стандартных, обеспечивают уменьшение модуля проекции градиента, за счет чего точки, находящиеся на большом удалении от стационарных точек функции, исключаются из рассматриваемых, что позволяет ускорить работу метода и избежать возникновение бесконечного цикла при линейном поиске в тех случаях, когда для обычных Wolfe Conditions при выполнении первого из условий во втором достигается равенство, и на дальнейших итерациях изменения параметров столь малы, что равенство с точки зрения используемого ПО (его ограничений на точность представления чисел типа *double*) не нарушается.

Примеры работы:



Восстановление функции

$$y = -0.5 \cdot cos(x) + 3.19 \cdot log(x) + 3.03 \cdot log^3(x) + 2.26 \cdot sin^2(x)$$

L-BFGS method

L-BFGS (Limited memory BFGS) - метод, являющийся модификацией алгоритма BFGS, позволяющий сократить потребляемую при исполнении память ($\mathcal{O}(n^2)$ complexity in BFGS) за счет хранения, вместо Гессиана, наборов (в текущей реализации использована структура Deque) значений параметров $\{\rho_k\}_{k=0}^{m-1}$, $\{s_k\}_{k=0}^{m-1}$, $\{\alpha_k\}_{k=0}^{m-1}$, $\{y_k\}_{k=0}^{m-1}$, полученных на предыдущих m итерациях метода.

$s_i = x_{i+1} - x_i = \nabla f_{i+1} - \nabla f_i$

α_i - значение коэффициента α , найденное на i-й итерации

$y_i = f(x_i)$

$\rho_i = \dfrac{1}{(y_0, s_0) + \varepsilon \cdot 10^{-3}}$ - тут индексы у y_0 , s_0 считаются относительно начала очереди.

В данном случае $x_{k+1} = x_k - \alpha_k p_k$

Значение $p_k = H_k \nabla f_k$ в условиях отсутствия explicit inverse Hessian вычисление p_k осуществляется следующим образом:

$for(int\ i = 0; i < m; i\ ++)\{$

$\alpha = \rho_i \cdot (s_i, q);$

$q- = y_i \cdot \alpha;$

}

$\gamma = \dfrac{(s_0, y_0)}{(y_0, y_0) + \varepsilon \cdot 10^{-3}};$

$p = q \cdot \gamma;$

$for(int\ i = m - 1; i \geq 0; i\ --)\{$

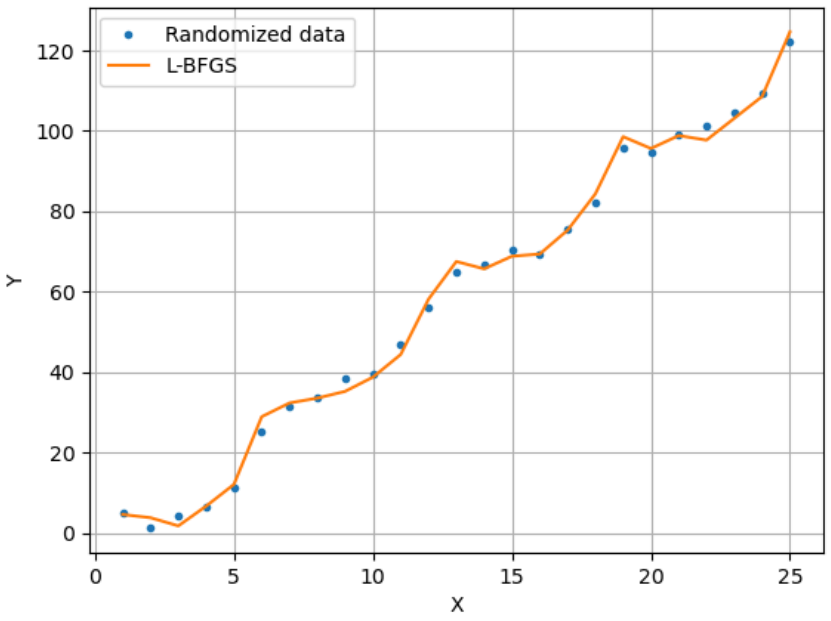
$\beta = \rho_i \cdot (y_i, p);$

$p+ = s_i \cdot (\alpha_i - \beta);$

}

С конкретной реализацией можно ознакомиться по ссылке.

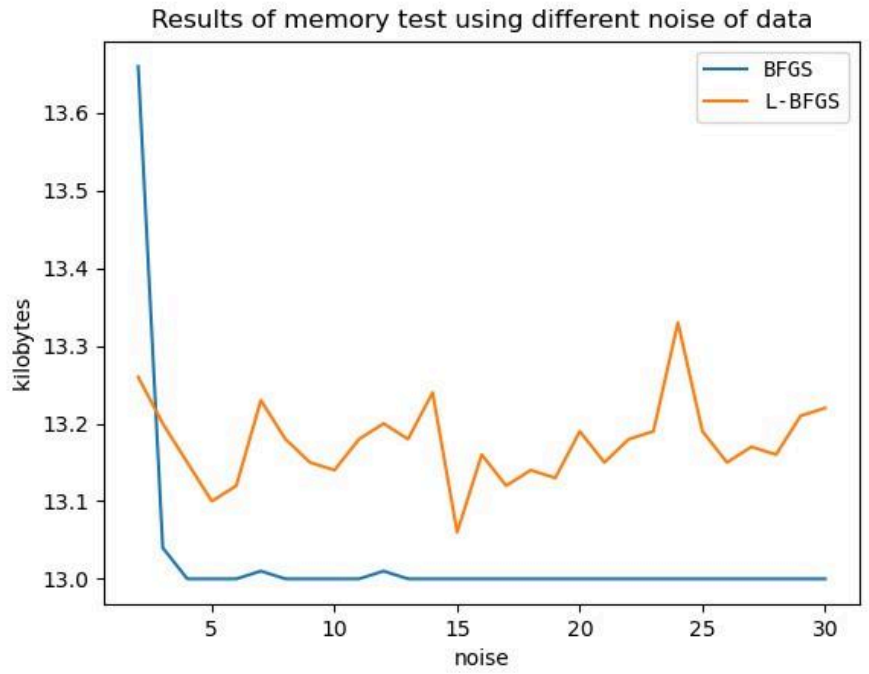
Примеры работы:



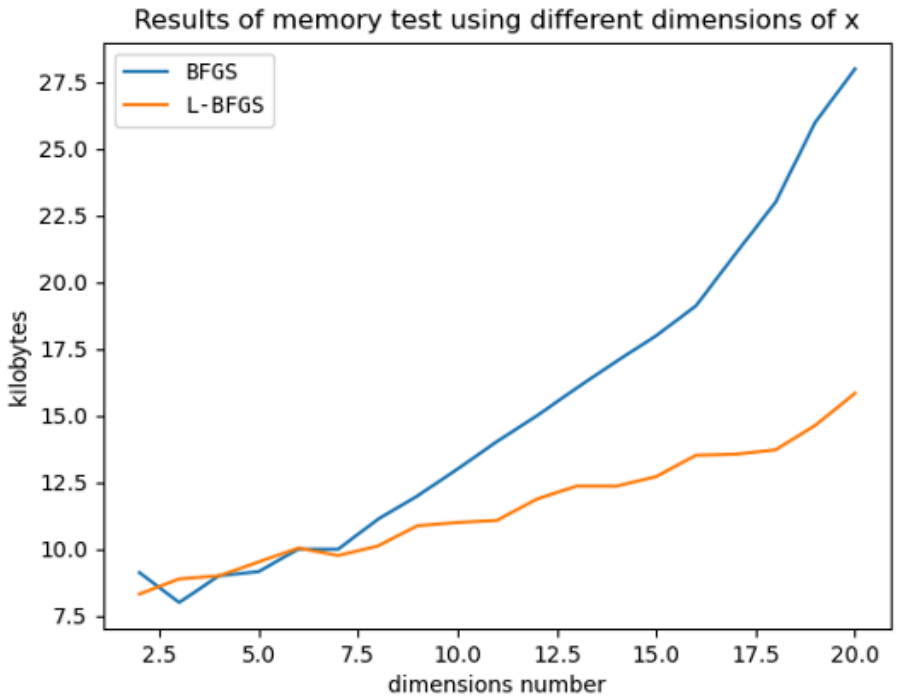
Восстановление функции

$y = 7.22 \cdot cos^3(x) + 3.42 \cdot log^2(x) + 2.76 \cdot sin(x) + 3.94 \cdot cos^2(x)$

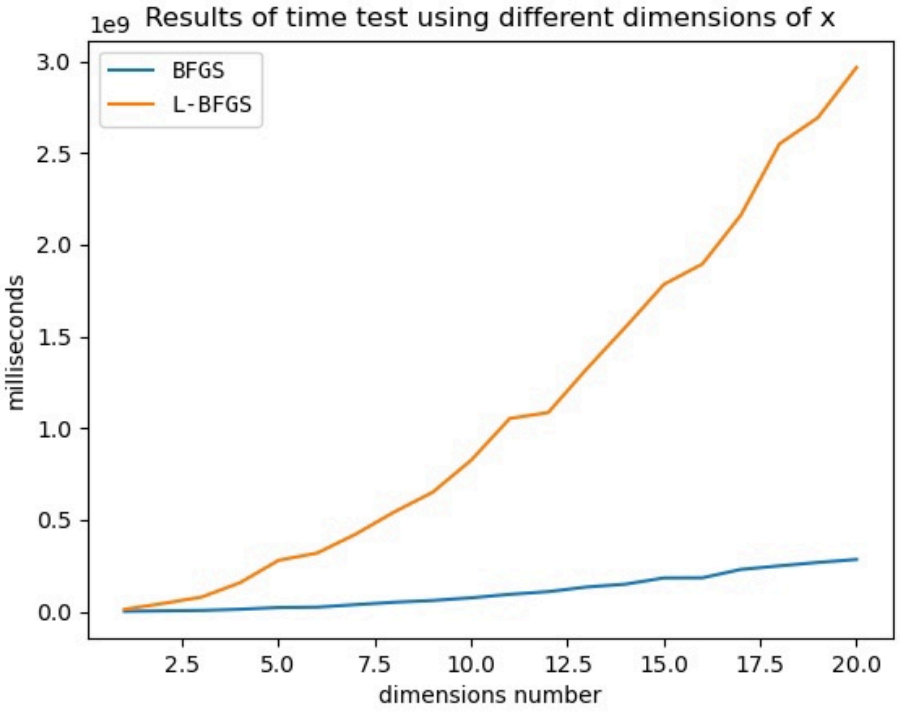
Сравнение BFGS и L-BFGS по памяти



Основные преимущества L-BFGS раскрываются при больших значениях измерений, что можно увидеть ниже.



Образцовый график превосходства L-BFGS над обычным BFGS. При небольшом количестве измерений количество потребляемой памяти сопоставимо, однако при увеличении числа измерений L-BFGS растёт гораздо медленнее.



Побочным эффектом уменьшения потребляемой памяти является рост времени работы.

Выводы

Мы реализовали следующие методы в этой лабораторной работе:

1. Метод Гаусса-Ньютона
2. Powell's Dog Leg
3. BFGS (Broyden-Fletcher-Goldfarb-Shanno algorithm)
4. L-BFGS (Limited-memory BFGS)

Каждый из алгоритмов показал улучшенные результаты по сравнению с алгоритмами из прошлых лабораторных работ. В некоторых случаях метод Гаусса-Ньютона показал себя лучше, чем Powell's Dog Leg.