

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

Методы оптимизации

Отчёт по лабораторной работе №4

Выполнили:
Ночевкина Наталья, М3233
Чуракова Александра, М3232
Ильченко Артём, М3233

Санкт-Петербург
2023

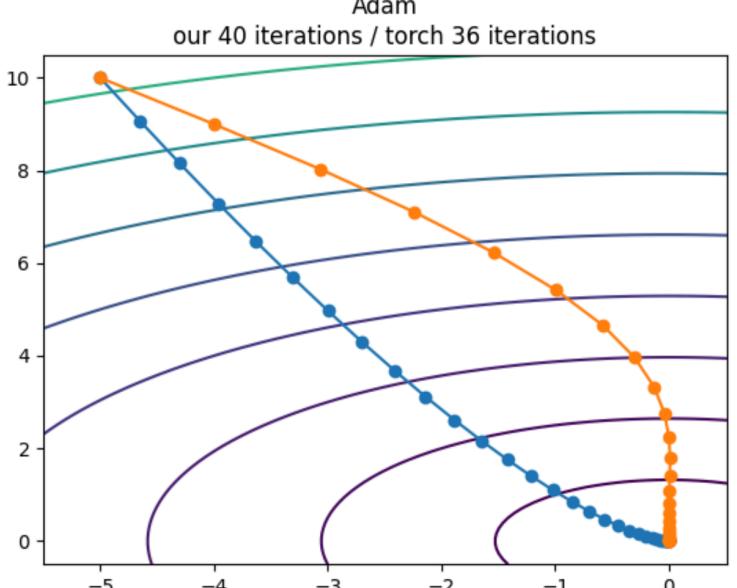
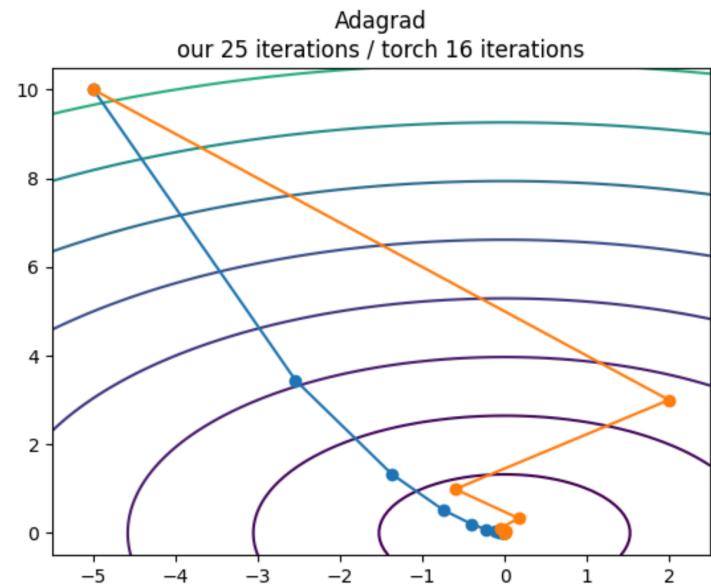
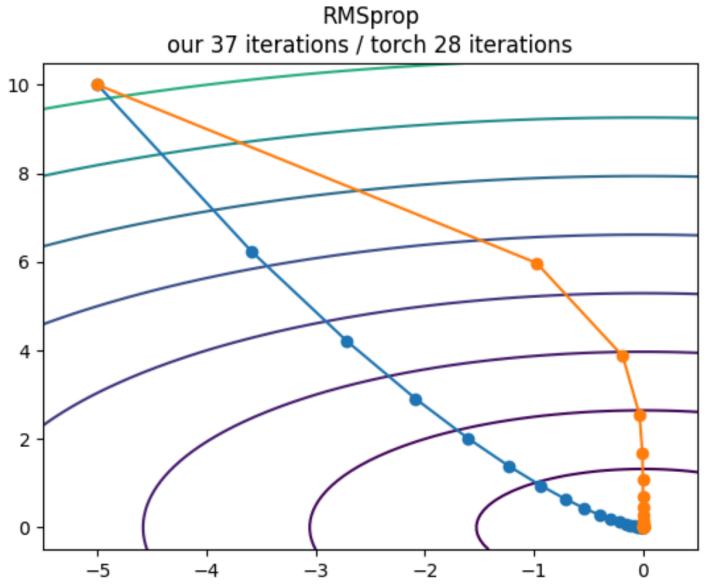
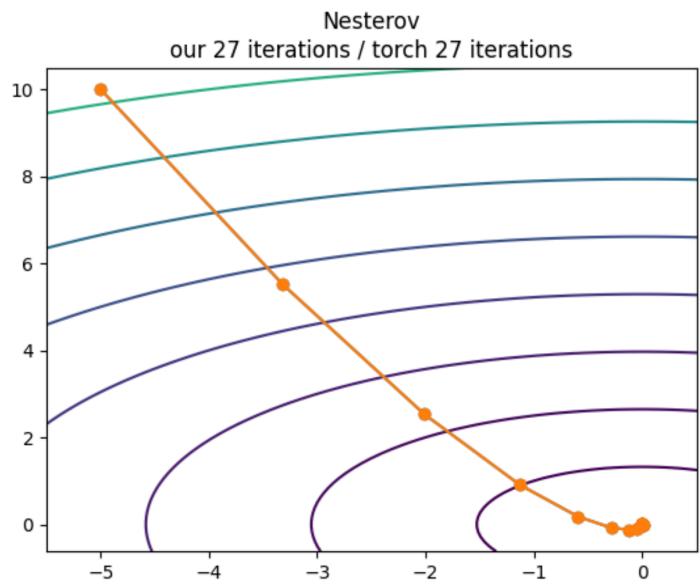
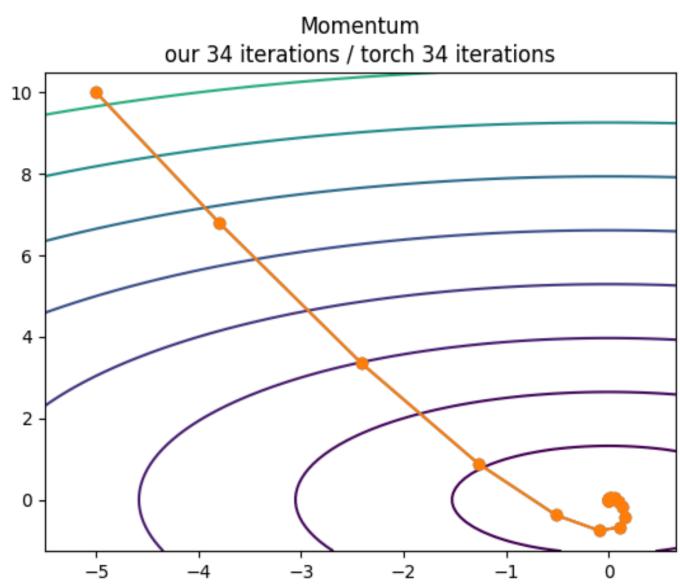
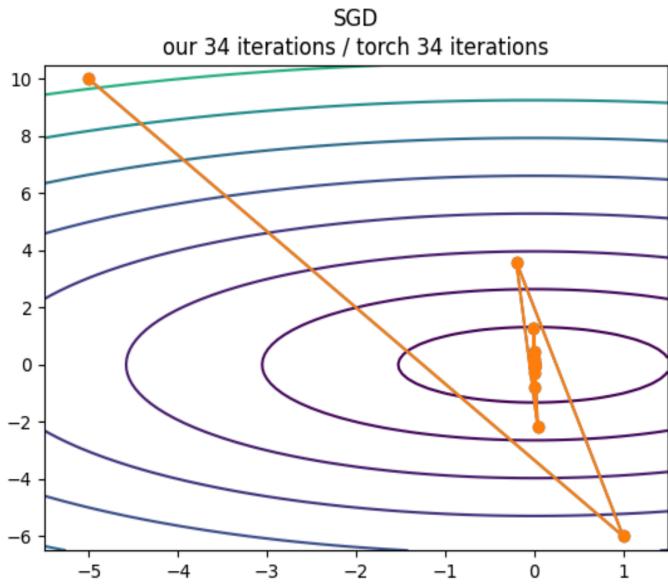
Цели работы

1. Изучить работу методов из библиотек `torch.optim` и `scipy.optimize`
2. Сравнить показатели эффективности “библиотечных” методов с показателями методов, реализованных в прошлых работах
3. Исследовать влияние ограничений на работу “библиотечных” методов

Задачи

1. Изучить и проанализировать работу `sgd`, реализованного в `torch.optim` и сравнить с собственными реализациами
2. Исследовать методы регрессии из библиотеки `scipy.optimize`, сравнив их показатели с показателями собственных имплементаций
3. Ознакомиться с методом вычисления градиента PyTorch и сравнить с другими используемыми подходами
4. Исследовать влияние границ изменения параметров на работу методов
5. Исследовать использование `LinearConstraint` и `NonLinearConstraint`, представленных в `scipy.optimize` на работу методов из этой библиотеки

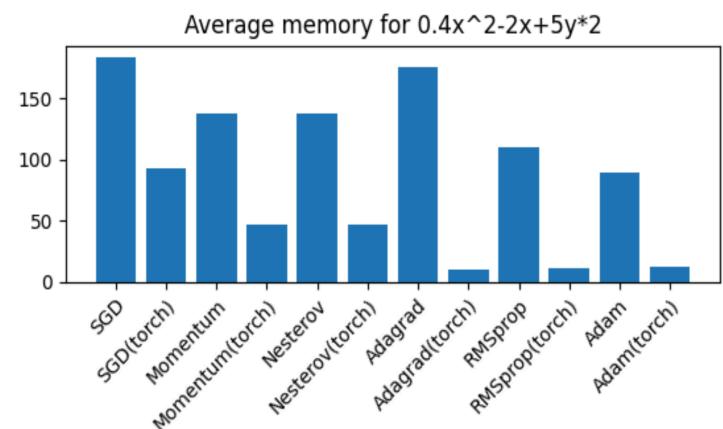
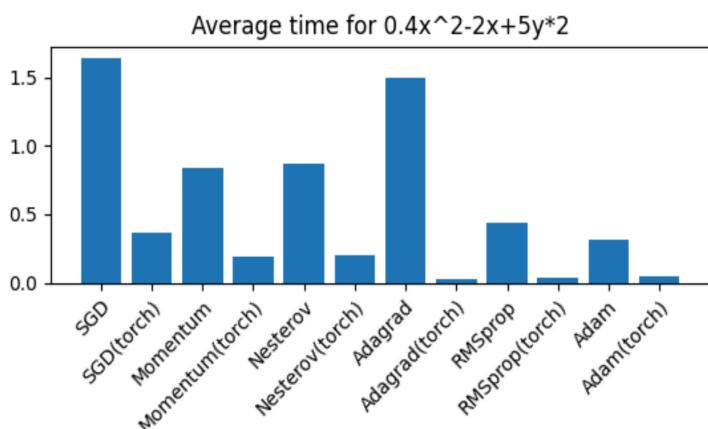
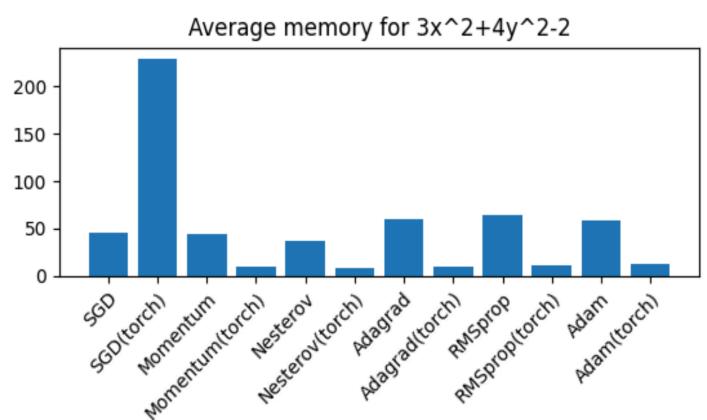
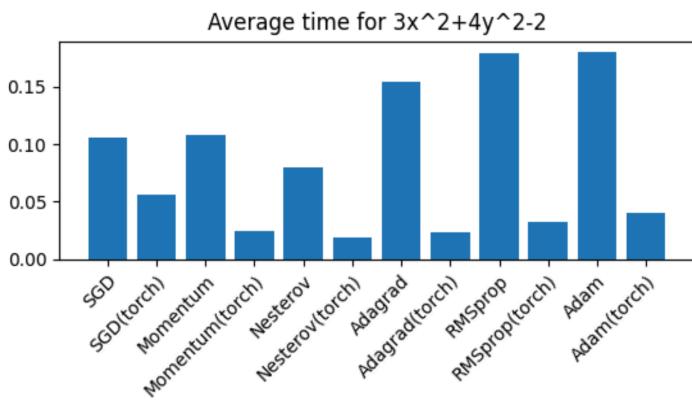
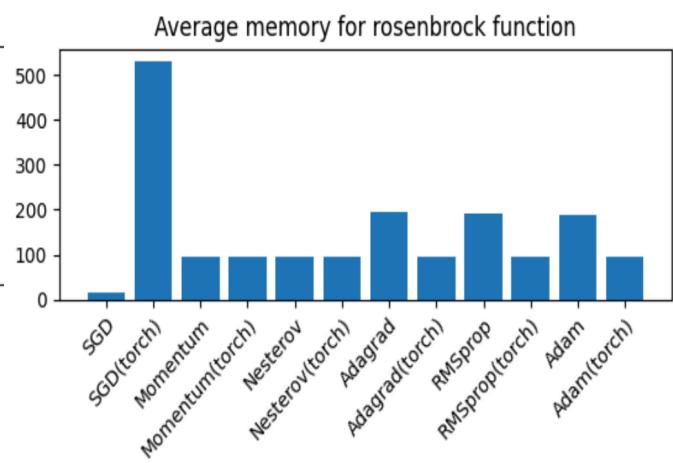
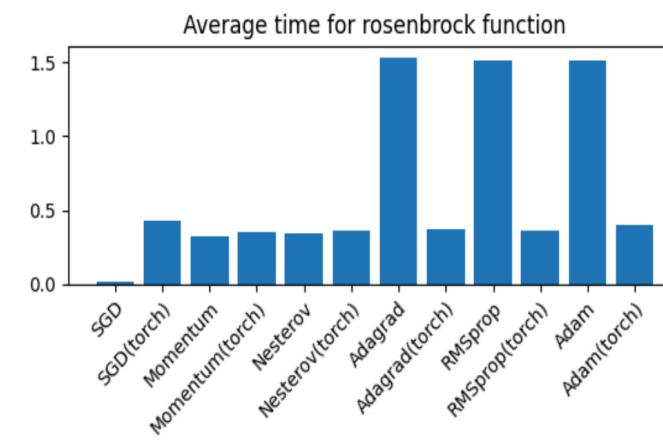
1. SGD

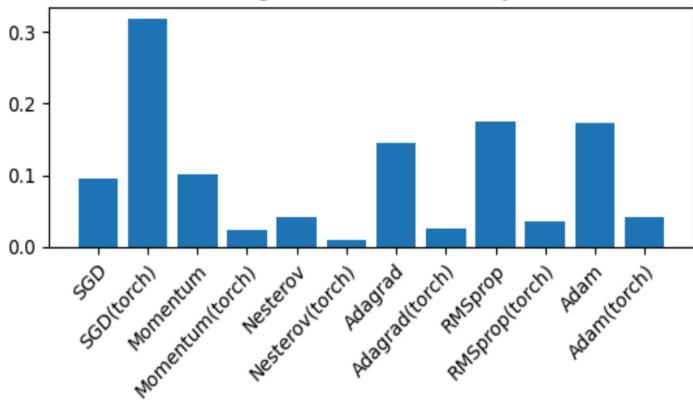
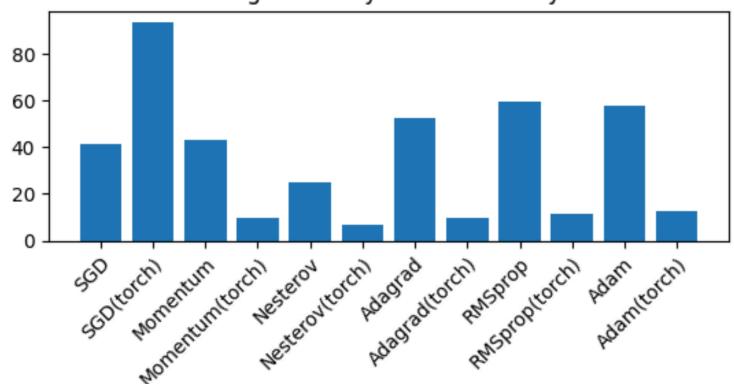
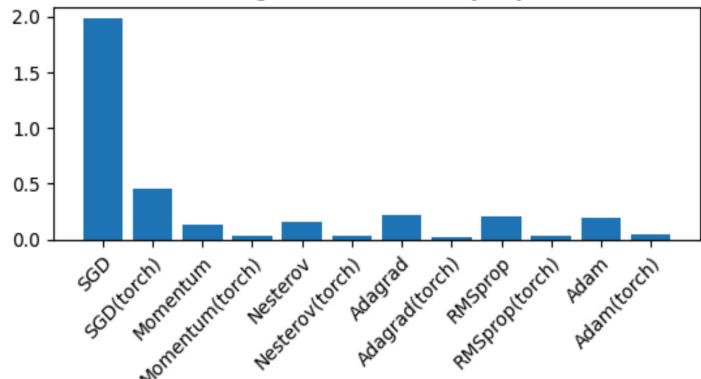
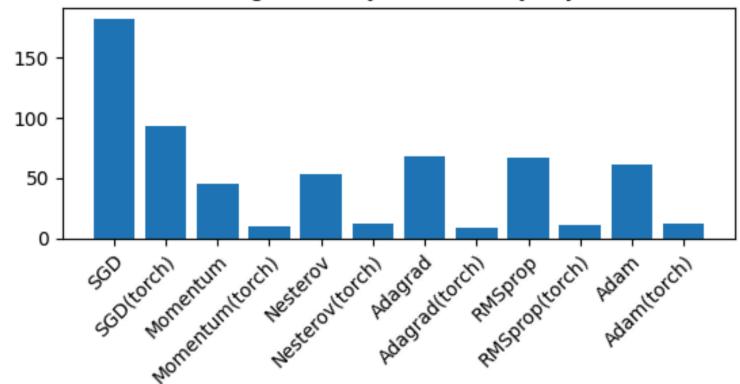


На изображениях выше приведены примеры работы наших имплементаций SGD, Momentum, NesterovGd, RMSProp, Adagrad, Adam вместе с работой тех же методов из библиотеки torch.optim при минимизации функции $3x^2 + 4y^2 - 2$

Траектория спуска библиотечных методов отображается оранжевым цветом, наших - синим.

Как видно, в некоторых случаях, как например, Momentum, Nesterov, SGD, реализованные нами методы полностью совпадают как по траектории, так и по количеству итераций с методами torch.optim. Adagrad, RMSProp, Adam при условии запуска на одинаковых параметрах несколько уступают по количеству итераций библиотечным, имея при этом более плавную траекторию спуска.



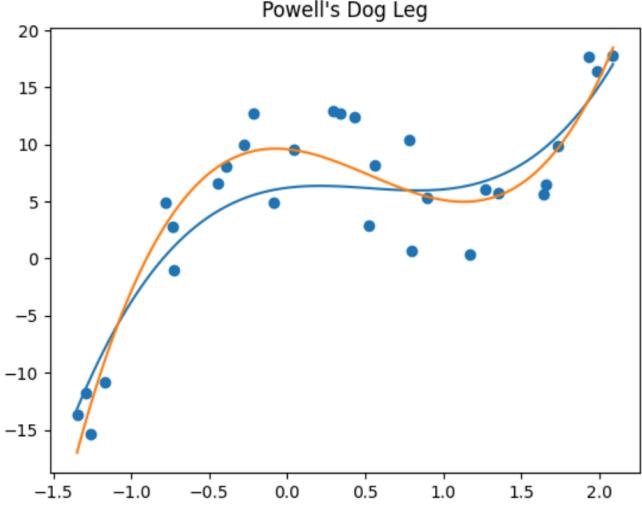
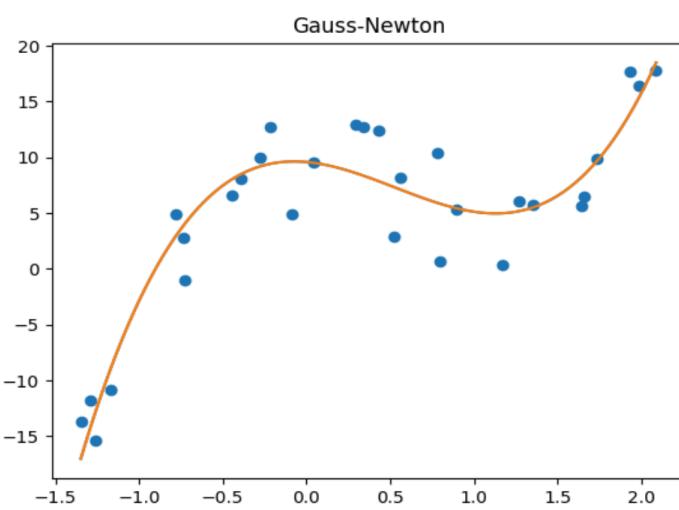
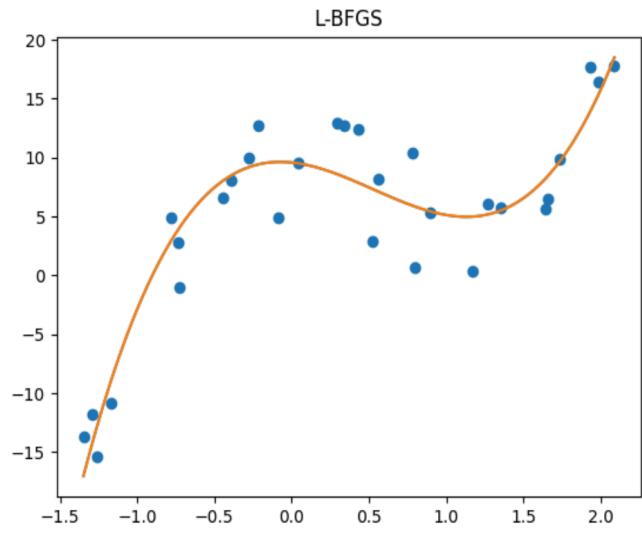
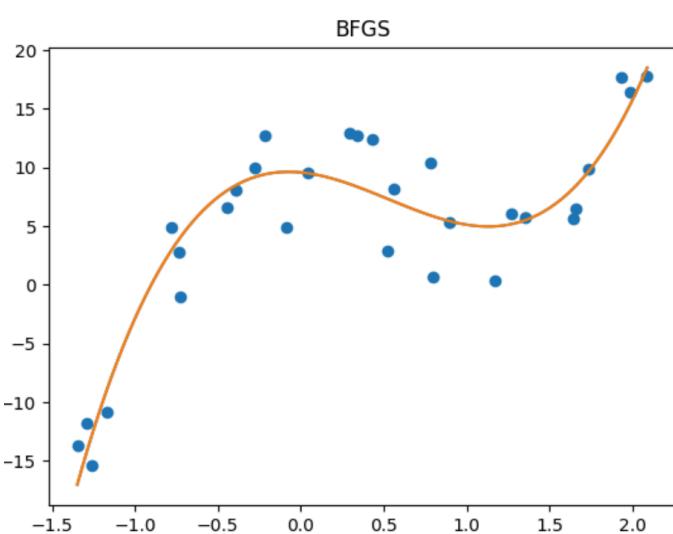
Average time for $10x^2+10y^2$ Average memory for $10x^2+10y^2$ Average time for $5x^2+xy+2y^2$ Average memory for $5x^2+xy+2y^2$ 

Для более детального сравнения эффективности методов были проведены замеры абсолютных значений времени и потребляемой памяти при работе наших и библиотечных методов. Ожидаемо, наши реализации сильно уступают библиотечным по этим показателям, что связано как с более профессиональной реализацией, так и, вероятно, с тем, что внутри методы из torch.optim работают на базе C++, который в целом сильно быстрее и эффективнее нежели python.

2. Scipy.optimize.minimize & least_squares

Для задачи восстановления многочлена с помощью регрессии нами в предыдущих лабораторных были реализованы методы с использованием алгоритмов BFGS, L-BFGS, а также Gauss-Newton method, Powell's DogLeg method, с сранение которым в данном случае сопоставляются bggs l-bfgs-b, least_squares, least-squares(dogbox), соответственно, первые два из которых относятся к `scipy.optimize.minimize`, а вторые к `scipy.optimize.least_squares`

Ниже приведены примеры восстановления полинома $5x^3 - 8x^2 - x + 9$ нашими и библиотечными методами(графики полиномов, полученных с помощью наших имплементаций отображаются синим цветом, с помощью `scipy.optimize` - оранжевым)

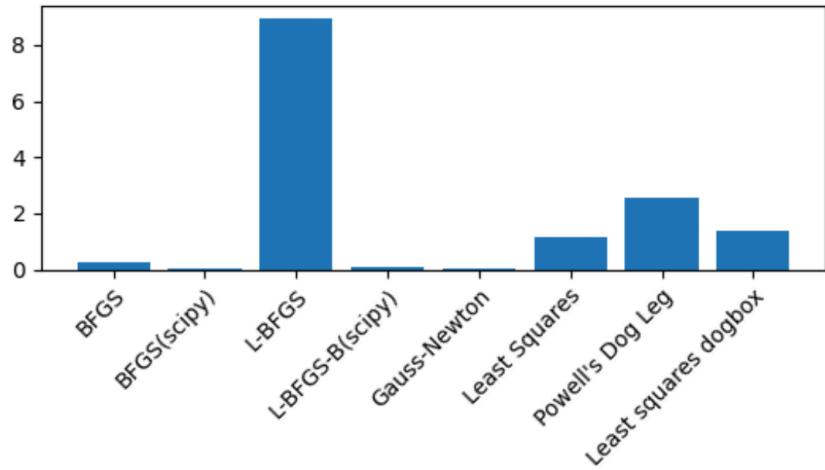


Как видно по изображениям и библиотечные, и наши методы достаточно хорошо справляются с задачей поиска коэффициентов, а Powell's Dog Leg даже более точен, чем least-squares(dogbox) из scipy

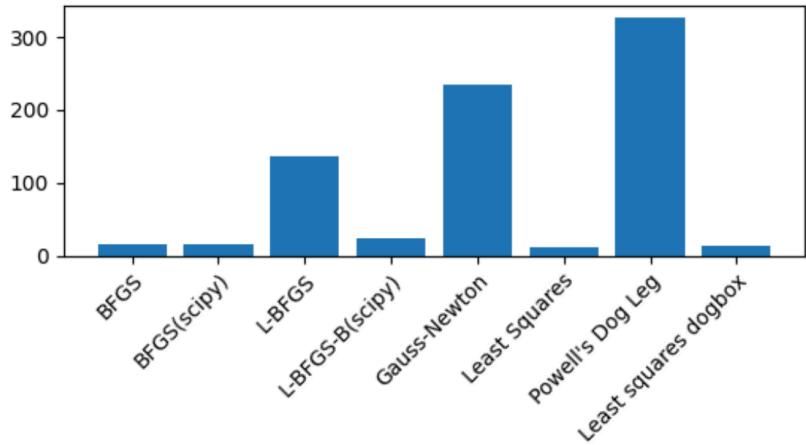
Далее приводятся показатели работы методов на функциях

- $\log 5x^4 - e^{x^{\frac{3}{2}}} - 2x^2 + 4x - 5$
- $\frac{e^{x^2}}{8x^5 - 9}$
- $8x \cdot e^{\sin x^4} + 3x^2$

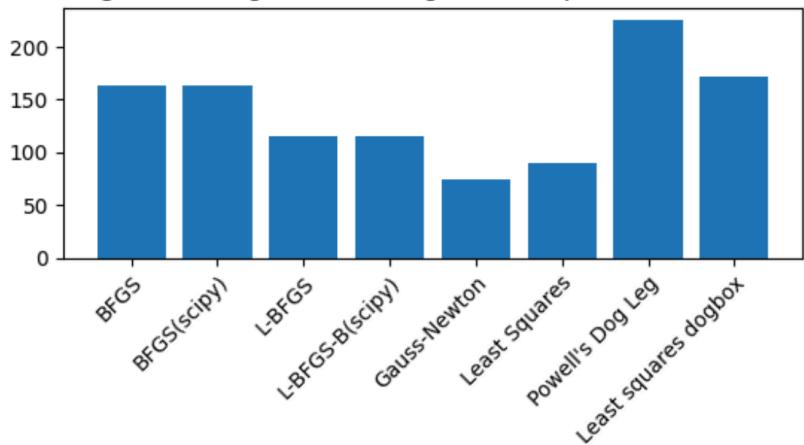
Average time regression for $\log(5x^4) - \exp(x)^{1.5} - 2x^2 + 4x - 5$



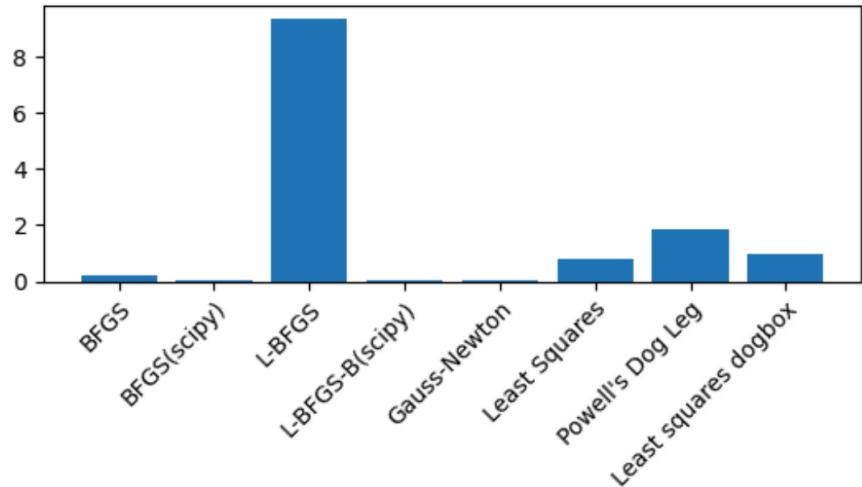
Average memory regression for $\log(5x^4) - \exp(x)^{1.5} - 2x^2 + 4x - 5$



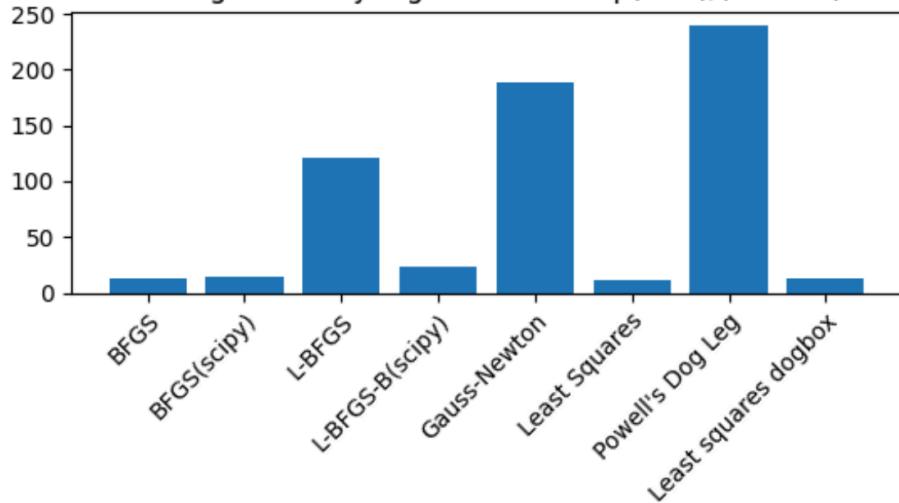
Average losses regression for $\log(5x^4) - \exp(x)^{1.5} - 2x^2 + 4x - 5$



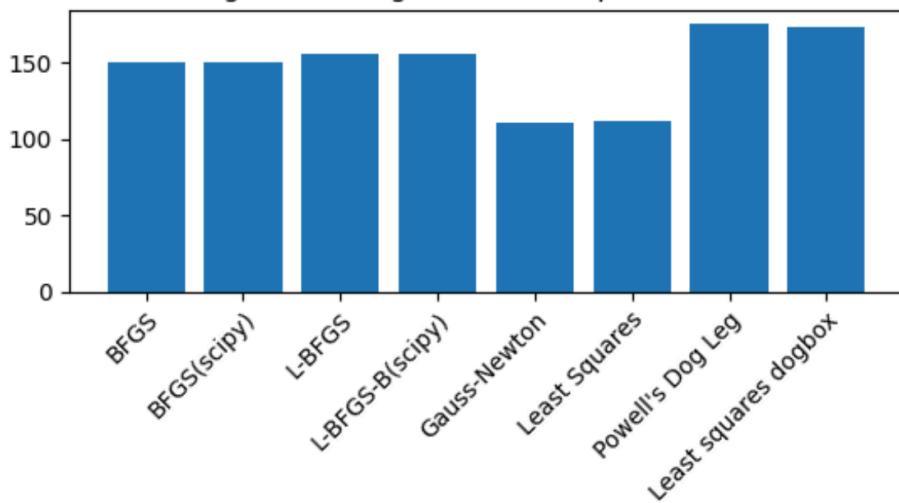
Average time regression for $\exp(x^2)/(8x^{5-9})$



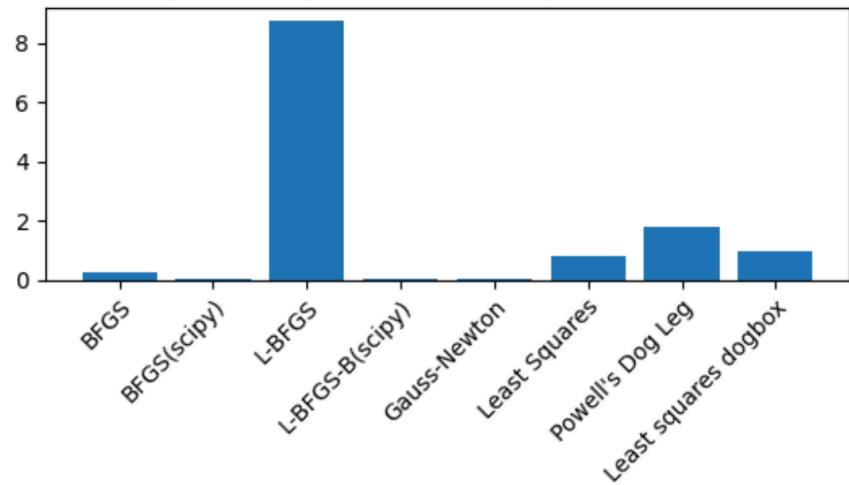
Average memory regression for $\exp(x^2)/(8x^{5-9})$



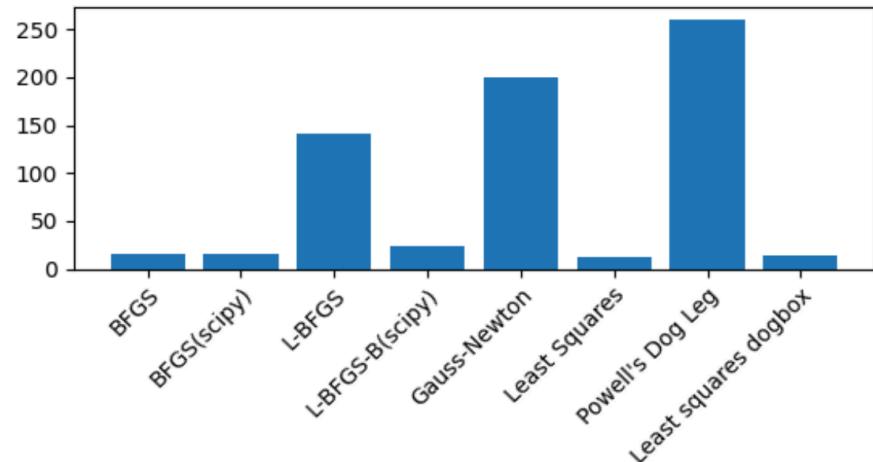
Average losses regression for $\exp(x^2)/(8x^{5-9})$



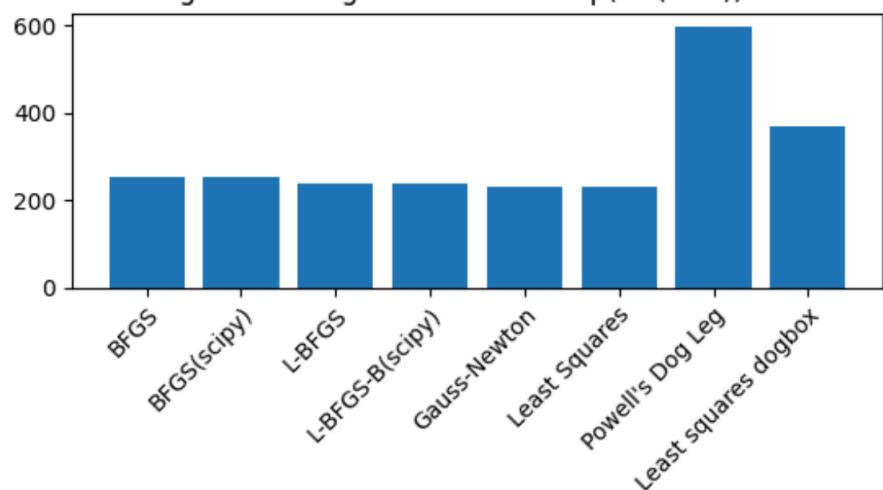
Average time regression for $8x \cdot \exp(\sin(x^4)) + 3 \cdot x^2$



Average memory regression for $8x \cdot \exp(\sin(x^4)) + 3 \cdot x^2$



Average losses regression for $8x \cdot \exp(\sin(x^4)) + 3 \cdot x^2$

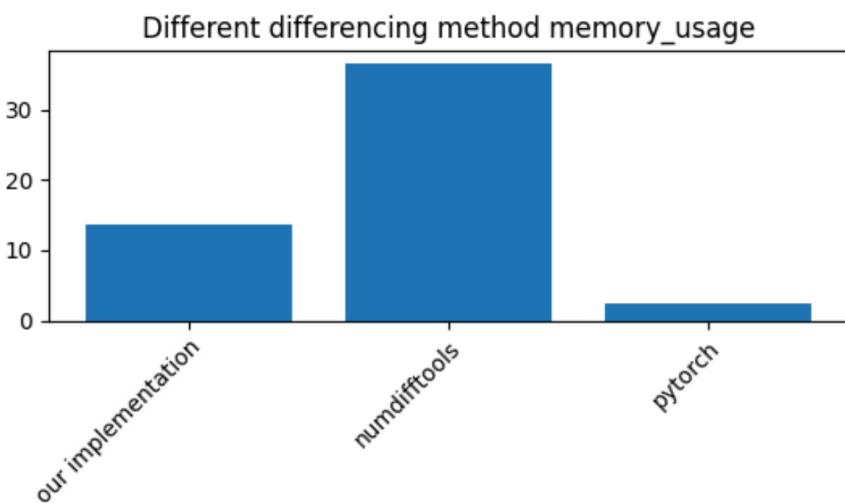
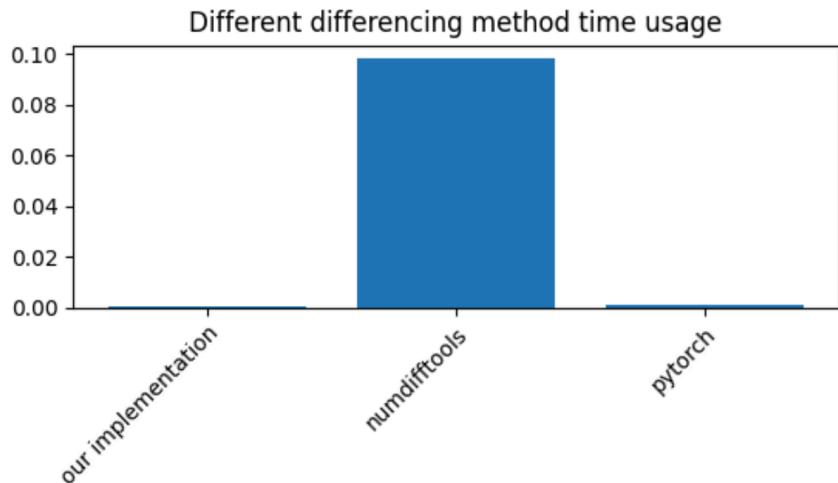


Проведенные исследования показывают, что наши методы не уступают по точности методам из `scipy.optimize`, однако значительно проигрывают им по показателям потребляемой памяти. Существенные различия во времени работы наблюдаются для методов BFGS и L-BFGS, в нашей имплементации проигрывающих библиотечным; Gauss Newton и Powell's Dog Leg также уступают `least_squares`, но не столь значительно. Тем не менее, выше есть несколько случаев, в котором наша реализация Gauss-Newton обогнала `least_squares` по результатам.

b Сравнение методов по вычислению градиента

Были исследованы 3 метода вычисления градиента:

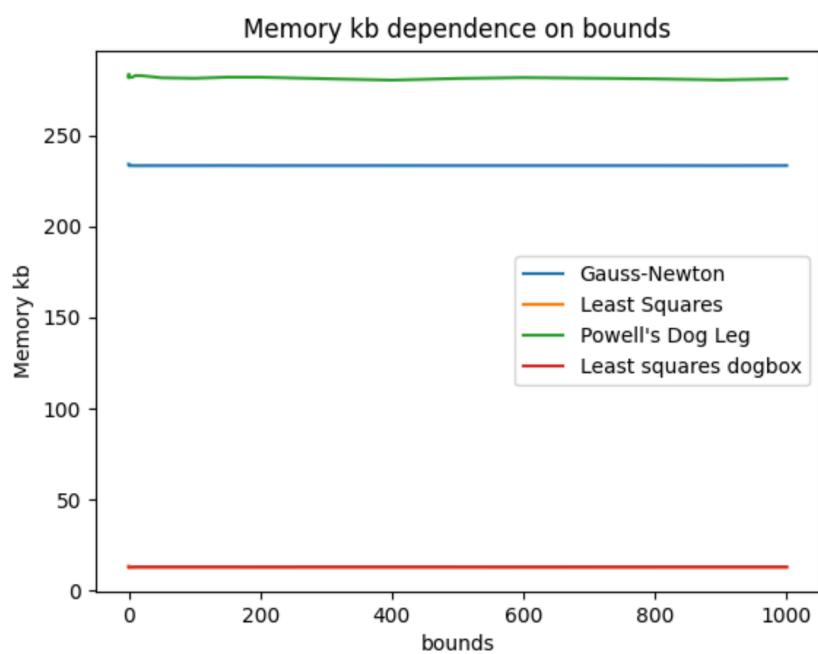
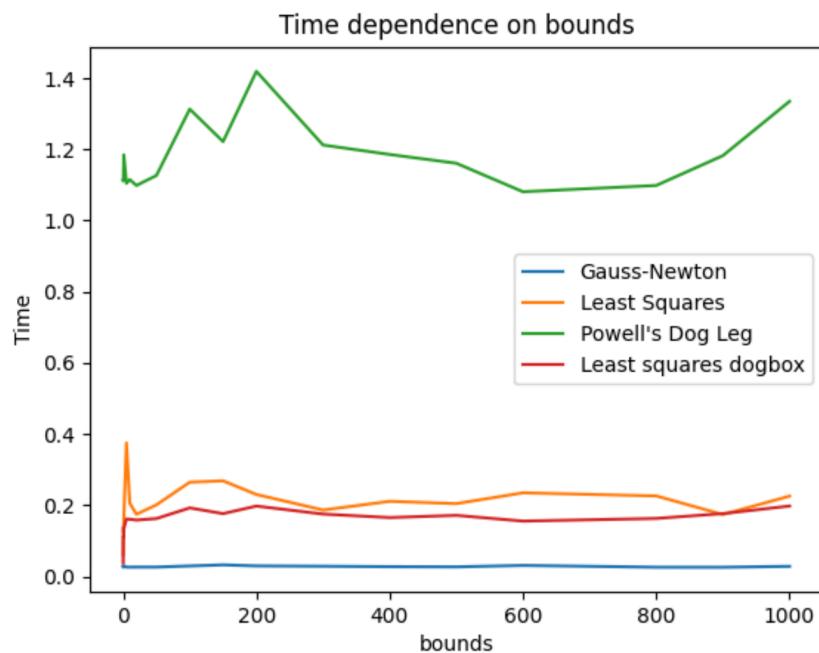
- численный метод, реализованный самостоятельно (можно ознакомиться по [ссылке](#))
- [ndt.Gradient](#) из библиотеки numdifftools
- вычисление градиента с помощью pytorch

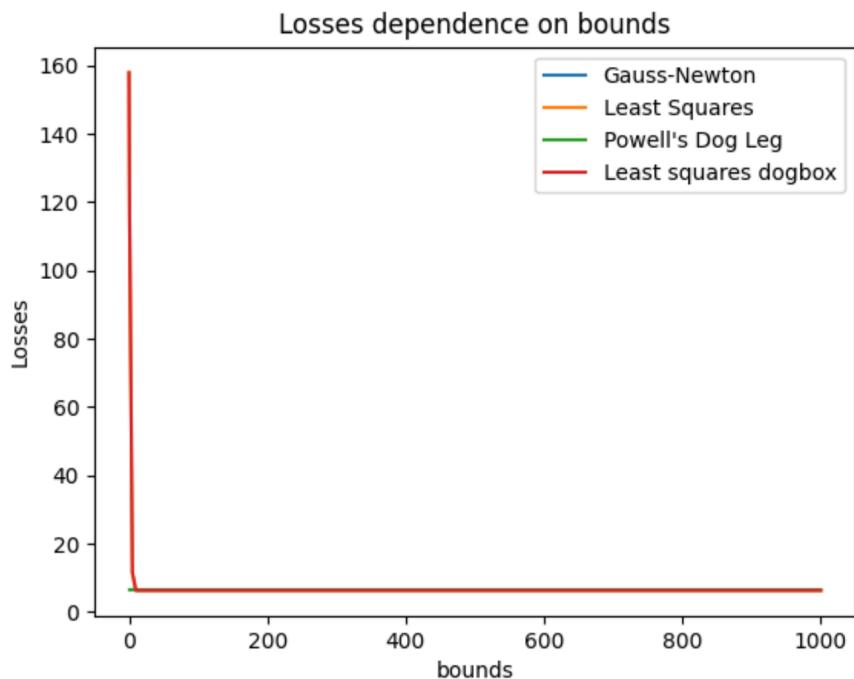


Наиболее эффективным показал себя pytorch. Численный метод вычисления градиента лишь незначительно(близко к погрешности) превосходит pytorch по абсолютному времени работы, но ощутимо проигрывает по затрачиваемой памяти. Метод из numdifftools и по памяти, и по времени значительно проигрывает предыдущим двум. Что касается точности вычислений, то при проведении испытаний между методами не было обнаружено расхождений в вычисленных значениях градиентов.

С) Влияние границ

для исследования использовалась функция $2\sqrt{e^{x^3}} - x^3 + 4x^2 - 8$,
границы которой определялись отрезком $[-x; x]$,
 $x \in \{0.1, 0.25, 0.5, 1.0, 5.0, 10.0, 20.0, 50.0, 100.0, 150.0, 200.0, 300.0, 400.0, 500.0, 600.0, 800.0, 900.0, 1000.0\}$





Как видно худшие показатели появляются при слишком узких границах. Расширение интервала после преодоления 1 сначала положительно влияет на время, память и значения функции потерь. При достаточно широком интервале постепенно начинает увеличиваться время работы методов. На память и размер ошибки задаваемые значения границ не оказывают существенного влияния.

Доп задание

Исследование использования линейных и нелинейных ограничений:

Constraint	Time, sec	Memory, kb	Iterations	Function	x
\emptyset	0.156	25.33	177	$100(y - x^2)^2 + (1 - x^2)$	[0.999, 0.999]
$y \geq 0.5 \cdot x$	0.064	19.69	67	$100(y - x^2)^2 + (1 - x^2)$	[0.054, 0.027]

$y \leq 2 \cdot x$	0.031	17.79	34	$100(y - x^2)^2 + (1 - x^2)$	[0.999, 0.999]	
$y \leq x$	0.033	17.78	36	$100(y - x^2)^2 + (1 - x^2)$	[0.999, 0.999]	
$y \geq x$	0.101	17.80	100	$100(y - x^2)^2 + (1 - x^2)$	[5.550, 30.809]	
$y = x$	0.018	17.10	18	$100(y - x^2)^2 + (1 - x^2)$	[1.000, 1.000]	
$B((1, 1), 1)$ круг с радиусом 1 от (1, 1)	0.055	18.32	37	$100(y - x^2)^2 + (1 - x^2)$	[0.999, 0.999]	
$B((0, 1), 1)$ круг с радиусом 1 от (0, 1)	0.059	20.12	44	$100(y - x^2)^2 + (1 - x^2)$	[0.999, 0.999]	
$B((0, 0), 1)$ круг с радиусом 1 от (0, 0)	0.059	20.10	45	$100(y - x^2)^2 + (1 - x^2)$	[0.786, 0.617]	
$\text{not } B((1, 1), 1)$ вне круга с радиусом 1 от (1, 1)	0.111	19.94	75	$100(y - x^2)^2 + (1 - x^2)$	[0.428, 0.179]	
$\text{not } B((0, 0), 1)$ вне круга с радиусом 1 от (0, 0)	0.082	20.00	61	$100(y - x^2)^2 + (1 - x^2)$	[-0.783 ,	0.620]
\emptyset	0.012	14.22	16	$(x - 1)^2 - 7xy + (y - 1)^4$	[14.872 ,	3.963]
$y \geq 0.5 \cdot x$	0.012	17.70	11	$(x - 1)^2 - 7xy + (y - 1)^4$	[7.378, 3.689]	
$y \leq 2 \cdot x$	0.012	17.75	11	$(x - 1)^2 - 7xy + (y - 1)^4$	[14.872 ,	3.963]
$y \leq x$	0.012	17.75	11	$(x - 1)^2 - 7xy + (y - 1)^4$	[14.872 ,	3.963]
$y \geq x$	0.015	17.75	15	$(x - 1)^2 - 7xy + (y - 1)^4$	[3.151, 3.151]	

$y = x$	0.014	17.08	15	$(x - 1)^2 - 7xy + (y - 1)^4$	[3.151, 3.151]
$B((1, 1), 1)$ круг с радиусом 1 от (1, 1)	0.040	20.48	28	$(x - 1)^2 - 7xy + (y - 1)^4$	[1.707, 1.707]
$B((0, 1), 1)$ круг с радиусом 1 от (0, 1)	0.036	20.53	21	$(x - 1)^2 - 7xy + (y - 1)^4$	[0.879, 1.476]
$B((0, 0), 1)$ круг с радиусом 1 от (0, 0)	0.038	20.51	23	$(x - 1)^2 - 7xy + (y - 1)^4$	[0.722, 0.691]
$\text{not } B((1, 1), 1)$ вне круга с радиусом 1 от (1, 1)	0.040	20.33	23	$(x - 1)^2 - 7xy + (y - 1)^4$	[14.871 ,3.963]
$\text{not } B((0, 0), 1)$ вне круга с радиусом 1 от (0, 0)	0.025	20.33	15	$(x - 1)^2 - 7xy + (y - 1)^4$	[-0.974 ,-0.225]

Линейные ограничения требуют меньше памяти и времени, чем нелинейные, и при сокращении области поиска сокращается и время работы метод, причем положение точки минимума внутри области поиска или на ее границе не оказывает существенного влияния на скорость сходимости и точность.

Если задать область, перекрывающую искомую точку, алгоритм не сойдется.