Методы онлайн-выпуклой оптимизации с ограничениями: сравнение Polyak Feasibility Steps, проекционных и primal–dual алгоритмов

Олег Бациев, Ксения Карпеева, Александра Иванова,

Innopolis University
22 декабря 2025

Аннотация

В работе исследуются методы онлайн-выпуклой оптимизации с ограничениями. Основное внимание уделено сравнению подхода Polyak Feasibility Steps (PFS), предложенного в недавней работе 2025 года, с классическими алгоритмами, используемыми в constrained OCO.

В рамках единого экспериментального протокола реализованы и сопоставлены четыре метода: онлайн-градиентный спуск с шагами допустимости Поляка (PFS), primal–dual алгоритм Drift-Plus-Penalty (DPP), его модификация с затянутым ограничением (DPP-T), а также проекционный онлайн-градиентный метод (POGD).

Экспериментальное сравнение проведено на двух типичных задачах: синтетической квадратичной задаче с простыми ограничениями и задаче онлайн-логистической регрессии с ограничением на норму параметров. Анализируются как показатели качества оптимизации (regret), так и характеристики соблюдения ограничений.

Содержание

1 Введение и мотивация

Задачи онлайн-оптимизация возникают в ситуациях, где решения необходимо принимать последовательно, а стоимость ошибки становится известна только после действия. Типичные примеры включают онлайн-обучение моделей, адаптивную настройку систем, распределение ресурсов, рекламные ставки и рекомендательные системы. Формально такие задачи описываются моделью Online convex optimization (OCO), в которой на шаге t алгоритм сначала выбирает решение x_t, а затем наблюдает выпуклую функцию потерь $f_t(\cdot)$ и несёт потерю $f_t(x_t)$ [1, 5].

Во многих прикладных задачах, помимо минимизации потерь, присутствуют ограничения: бюджеты, ограничения на риск и нормы параметров, требования безопасности или fairness. В этом случае важно не только уменьшать суммарные потери, но и *контролировать нарушения ограничений во времени*, что приводит к постановке задачи онлайн-выпуклой оптимизации с ограничениями (constrained OCO) [1].

1.1 Формальный пример задачи

Рассмотрим задачу онлайн-оптимизации, соответствующую второму бенчмарку, а именно онлайн-логистической регрессии с ограничением на норму параметров. Пусть $w \in \mathbb{R}^d$ — вектор параметров линейного классификатора. Процесс взаимодействия с окружающей средой происходит в течение T раундов. На каждом шаге $t = 1, \dots, T$ выполняются следующие действия:

1. алгоритм выбирает вектор параметров $w_t \in \mathbb{R}^d$;
2. после этого становится доступным наблюдение (a_t, b_t), где $a_t \in \mathbb{R}^d$ — вектор признаков, а $b_t \in \{-1, +1\}$ — бинарная метка;
3. алгоритм несёт значение функции потерь, заданной логистической функцией с ℓ_2-регуляризацией:

$$f_t(w) = \log\bigl(1 + \exp(-b_t\, a_t^\top w)\bigr) + \frac{\lambda}{2}\|w\|_2^2;$$

4. выбранные параметры должны удовлетворять выпуклому ограничению на норму

$$g(w) = \|w\|_2 - B \le 0, \qquad \text{эквивалентно } \|w\|_2 \le B.$$

Таким образом, задача заключается в последовательной минимизации суммарной логистической потери при наличии жёсткого ограничения на допустимую область параметров. Ограничение на ℓ_2-норму служит для предотвращения неограниченного роста параметров модели, что особенно важно в условиях шума, нестационарности или адверсариального характера поступающих данных.

1.2 Почему Constrained OCO является нетривиальной задачей

В отсутствие ограничений задача онлайн-оптимизации может быть решена стандартными методами первого порядка, такими как онлайн-градиентный спуск. Однако при наличии функциональных ограничений вида $g(w) \le 0$ ситуация существенно усложняется. В этом случае возникает принципиальный выбор между двумя альтернативными стратегиями:

- поддерживать допустимость решения на каждом шаге алгоритма, например, посредством проекции на допустимое множество;
- допускать временные нарушения ограничений, контролируя их лишь в среднем или в суммарном виде.

В существующей литературе по constrained online convex optimization можно выделить два основных класса методов.

Первый класс составляют проекционные методы, такие как Projected Online Gradient Descent. В этих алгоритмах после выполнения градиентного шага решение явно проецируется на истинное допустимое множество X. Такой подход обеспечивает строгую допустимость на каждом раунде, однако требует вычисления проекции на X, что может быть вычислительно затратным или даже практически неосуществимым, если множество X имеет сложную геометрию.

Второй класс методов включает примально-двойственные алгоритмы и методы с виртуальными очередями, в частности Drift-Plus-Penalty (DPP). В этих подходах ограничения учитываются косвенно, через множители Лагранжа или вспомогательные динамические переменные. Это позволяет избежать явных проекций на допустимое множество и зачастую приводит к более эффективной минимизации суммарной функции потерь. В то же время такие методы, как правило, не гарантируют строгую допустимость на каждом шаге и допускают положительные значения $g(x_t)$ на отдельных итерациях, контролируя лишь их суммарный рост во времени [2].

1.3 Идея Polyak Feasibility Steps

В недавней работе предложен метод Polyak Feasibility Steps (PFS) для решения задач constrained online convex optimization. Основная идея метода заключается в следующем. После выполнения стандартного онлайн-градиентного шага по функции потерь алгоритм применяет дополнительный корректирующий шаг, направленный на восстановление допустимости решения.

Этот корректирующий шаг имеет форму шага Поляка и использует значение функции ограничения $g(x)$, а также её субградиент в текущей точке. В отличие от проекционных методов, PFS не требует явного вычисления проекции на истинное допустимое множество X. Вместо этого корректировка осуществляется за счёт локальной информации о нарушении ограничения, что позволяет эффективно сдвигать текущую точку в сторону допустимой области даже при сложной геометрии множества X [3].

1.4 Цель проекта

Целью данной работы является построение воспроизводимого экспериментального стенда для сравнительного анализа алгоритмов онлайн-оптимизации с ограничениями. В рамках исследования рассматриваются четыре метода: Polyak Feasibility Steps (PFS), Drift-Plus-Penalty (DPP), его модификация с затянутым ограничением (DPP-T), а также Projected Online Gradient Descent (POGD).

Сравнение проводится на двух типах задач: синтетической квадратичной задаче и задаче онлайн-логистической регрессии. Качество работы алгоритмов оценивается по двум основным группам метрик: величине regret, характеризующей эффективность минимизации функции потерь, и степени нарушения ограничений, рассматриваемой как в покомпонентном (мгновенном), так и в кумулятивном виде.

2 Постановка задачи: OCO и Constrained OCO

В данном разделе фиксируются используемые обозначения и формально описывается постановка задачи. В дальнейшем при описании алгоритмов и экспериментальной части мы будем регулярно ссылаться на последовательности x_t, функции потерь f_t, величину regret и метрики нарушения ограничений.

2.1 Онлайн-выпуклая оптимизация (OCO)

Пусть d — размерность пространства решений (например, число параметров модели). Рассматривается процесс, состоящий из T раундов. На каждом шаге $t = 1, \ldots, T$ происходит следующий цикл:

1. алгоритм выбирает решение $x_t \in X$, где x_t является управляемой переменной (например, вектор параметров модели, действие агента или распределение ресурсов);

2. после выбора раскрывается функция потерь $f_t(\cdot)$, и алгоритм несёт потери в размере $f_t(x_t)$.

Ключевая особенность онлайн-постановки заключается в том, что функция потерь f_t неизвестна в момент выбора x_t и становится доступной только после принятия решения.

Выпуклость функций потерь. Предполагается, что каждая функция $f_t : \mathbb{R}^d \to \mathbb{R}$ является выпуклой. Это допущение позволяет применять градиентные и субградиентные методы оптимизации и получать теоретические гарантии сходимости.

Regret как метрика качества. Классической метрикой качества в задачах онлайн-выпуклой оптимизации является regret относительно лучшего фиксированного решения, выбранного задним числом:

$$\mathrm{Regret}(T) = \sum_{t=1}^{T} f_t(x_t) \; - \; \min_{x \in X} \sum_{t=1}^{T} f_t(x).$$

Интерпретация данной величины следующая:

- $\sum_{t=1}^{T} f_t(x_t)$ — суммарные потери онлайн-алгоритма вдоль его траектории;
- $\min_{x \in X} \sum_{t=1}^{T} f_t(x)$ — потери наилучшего фиксированного решения, выбранного с полным знанием всех функций потерь;
- regret измеряет, насколько онлайн-алгоритм проигрывает этому оптимальному апостериорному выбору.

Целью является достижение сублинейного роста regret, например $\mathrm{Regret}(T) = o(T)$, что эквивалентно условию $\mathrm{Regret}(T)/T \to 0$ при $T \to \infty$ [1].

Пример

Пусть $X = [-1, 1]$ и заданы функции потерь

$$f_1(x) = (x - 1)^2, \qquad f_2(x) = (x + 1)^2.$$

Если алгоритм выбирает $x_1 = 0$ и $x_2 = 0$, то

$$\sum_{t=1}^{2} f_t(x_t) = f_1(0) + f_2(0) = 1 + 1 = 2.$$

Оракул выбирает фиксированное $x \in [-1, 1]$, минимизируя

$$f_1(x) + f_2(x) = (x-1)^2 + (x+1)^2 = 2x^2 + 2,$$

минимум достигается при $x^\star = 0$, и его значение равно 2. Следовательно, regret в данном примере равен нулю.

2.2 OCO с ограничениями

Во многих прикладных задачах решения должны удовлетворять дополнительным ограничениям, таким как ограничения на норму параметров, бюджет или уровень риска. Предположим, что заданы:

- простое выпуклое множество X_0, на которое можно эффективно выполнять проекцию (например, шар, прямоугольник или симплекс);
- набор выпуклых ограничений $g_i(x) \le 0$, $i = 1, \dots, m$.

Тогда истинное допустимое множество имеет вид

$$X := \{x \in X_0 : \ g_i(x) \le 0, \ i = 1, \dots, m\}.$$

Роль множества X_0. Во многих алгоритмах constrained OCO (в частности, в DPP и PFS) используется проекция только на множество X_0, поскольку проекция на полное допустимое множество X может быть вычислительно сложной или недоступной. Ограничения g_i при этом учитываются косвенно, через штрафы, виртуальные очереди или корректирующие шаги.

Метрики нарушения ограничений. Так как некоторые алгоритмы допускают нарушение ограничений на отдельных шагах, вводятся количественные меры допустимости.

Мгновенное нарушение для одного ограничения определяется как

$$v_t := [g(x_t)]_+ = \max\{g(x_t), 0\}.$$

При наличии нескольких ограничений часто используется агрегированная мера, например

$$v_t := \sum_{i=1}^{m} [g_i(x_t)]_+ \quad \text{или} \quad v_t := \max_{i=1,\dots,m} [g_i(x_t)]_+,$$

в зависимости от целей анализа.

Кумулятивное нарушение за T шагов определяется как

$$V(T) = \sum_{t=1}^{T} v_t,$$

а *среднее нарушение* — как

$$\frac{V(T)}{T}.$$

Интерпретация данных величин следующая:

- если $v_t = 0$, то ограничение полностью выполнено на шаге t;
- если $V(T) = o(T)$, то ограничения в среднем удовлетворяются асимптотически;
- если $v_t \equiv 0$ для всех t, то допустимость соблюдается строго на каждом шаге.

Пример

Рассмотрим одно ограничение $g(x) = |x| - 0.5 \leq 0$, то есть $|x| \leq 0.5$.

Пусть алгоритм выбирает $x_1 = 0.4$, $x_2 = 0.8$, $x_3 = -0.6$. Тогда

$$v_1 = [|0.4| - 0.5]_+ = [-0.1]_+ = 0,$$

$$v_2 = [|0.8| - 0.5]_+ = [0.3]_+ = 0.3,$$

$$v_3 = [|-0.6| - 0.5]_+ = [0.1]_+ = 0.1.$$

Кумулятивное нарушение при $T = 3$ равно

$$V(3) = 0 + 0.3 + 0.1 = 0.4, \qquad \frac{V(3)}{3} \approx 0.133.$$

Две цели constrained OCO. В задачах Constrained online convex optimization обычно стремятся одновременно обеспечить:

1. малую величину regret по потерям;
2. малое нарушение ограничений (например, $V(T) = o(T)$).

Между этими требованиями, как правило, существует компромисс: усиление контроля допустимости часто приводит к ухудшению regret, и наоборот. В экспериментальной части работы алгоритмы сравниваются именно по этим двум критериям — качеству оптимизации и степени соблюдения ограничений [1, 2, 3].

3 Алгоритмы

В данном разделе описаны четыре алгоритма, реализованные и сравниваемые в проекте: проекционный онлайн-градиентный метод (POGD), примально-двойственный метод Drift-Plus-Penalty (DPP), его модификация с затянутым ограничением (DPP-T), а также онлайн-градиентный спуск с Polyak Feasibility Steps (PFS). Общее введение в онлайн-выпуклую оптимизацию и базовые методы приведено в обзоре Hazan [1], а методы с виртуальными очередями для учёта ограничений обсуждаются в работе Yu–Neely [2]. Метод PFS рассматривается в недавней статье 2025 года [3].

3.1 POGD: Projected Online Gradient Descent

Идея. POGD является базовым подходом к решению задач constrained OCO: после стандартного градиентного шага выполняется евклидова проекция на истинное допустимое множество X. В результате допустимость обеспечивается на каждом шаге, то есть мгновенные нарушения равны нулю.

Шаг алгоритма. На шаге t выполняются обновления

$$y_t = x_t - \eta_t \nabla f_t(x_t), \qquad x_{t+1} = \Pi_X(y_t),$$

где $\Pi_X(\cdot)$ обозначает евклидову проекцию на множество X.

Когда применяется.

- В случаях, когда проекция на X вычислительно проста (например, для box-ограничений, шара, симплекса или аффинных подпространств).
- В задачах, где требуется строгая допустимость на каждом шаге и нарушения ограничений недопустимы.

Преимущества.

- Строгая допустимость: $g(x_t) \leq 0$ для всех t, то есть $v_t = [g(x_t)]_+ = 0$.
- Простая интерпретация и надёжная базовая линия [1].

Недостатки.

- Проекция на X может быть вычислительно сложной, если множество задано неявно или имеет сложную геометрию.
- Для множеств вида $X = \{x \in X_0 : \ g(x) \leq 0\}$ вычисление Π_X может сводиться к решению отдельной вспомогательной задачи оптимизации.

Algorithm 1 POGD (Projected Online Gradient Descent)

1: **Input:** $x_t \in X$, шаг η_t
2: Получить градиент $g_t \leftarrow \nabla f_t(x_t)$
3: $y \leftarrow x_t - \eta_t g_t$
4: $x_{t+1} \leftarrow \Pi_X(y)$

3.2 DPP: Drift-Plus-Penalty (primal–dual, виртуальная очередь)

Идея. DPP относится к классу примально-двойственных методов для constrained OCO. Ограничение учитывается посредством виртуальной очереди (двойственной переменной) $Q_t \geq 0$. При нарушении ограничения величина Q_t возрастает и усиливает вклад ограничения в последующих обновлениях. Это позволяет избежать проекции на истинное множество X, ограничиваясь проекцией на простое множество X_0 [2].

Интерпретация через лагранжиан. Алгоритм можно интерпретировать как приближённую минимизацию на каждом шаге выражения

$$\mathcal{L}_t(x; Q_t) = f_t(x) + Q_t \, g(x),$$

с последующим обновлением Q_t, аккумулирующим нарушения ограничения.

Шаг алгоритма. Обновление решения имеет вид

$$x_{t+1} = \Pi_{X_0}\Big(x_t - \eta\big(\nabla f_t(x_t) + Q_t \nabla g(x_t)\big)\Big),$$

а обновление очереди задаётся правилом

$$Q_{t+1} = \max\{0,\ Q_t + c\, g(x_{t+1})\}.$$

Когда применяется.

- В задачах, где проекция на истинное множество X затруднена, но доступно простое множество X_0.
- В ситуациях, когда допускаются нарушения на отдельных шагах, однако требуется контроль суммарного нарушения $V(T)$.

Преимущества.

- Не требует проекции на X и использует только Π_{X_0}.
- На практике нередко демонстрирует хорошее качество по потерям за счёт допускаемых временных нарушений [2].

Недостатки.

- Мгновенная допустимость не гарантируется: величина v_t может быть положительной.
- Результат существенно зависит от выбора гиперпараметров (например, η и c), что может приводить либо к большим нарушениям, либо к ухудшению regret.

Algorithm 2 DPP (Drift-Plus-Penalty) [2]

Input: $x_t \in X_0$, $Q_t \geq 0$, шаг η, параметр очереди c
$g_t \leftarrow \nabla f_t(x_t)$
$s_t \leftarrow \nabla g(x_t)$
$x_{t+1} \leftarrow \Pi_{X_0}\big(x_t - \eta(g_t + Q_t s_t)\big)$
$Q_{t+1} \leftarrow \max\{0,\ Q_t + c\, g(x_{t+1})\}$

3.3 DPP-T: DPP с затянутым ограничением

Идея. В работе [3] рассматривается практический приём, заключающийся в замене ограничения $g(x) \leq 0$ на более жёсткое

$$g_\rho(x) = g(x) + \rho \leq 0, \qquad \rho > 0.$$

Такое затягивание формирует запас устойчивости: алгоритм стремится удерживать решения глубже внутри допустимой области, вследствие чего реальные нарушения $g(x) > 0$ возникают реже.

Как меняется DPP. В рамках данной модификации меняется правило обновления очереди:

$$Q_{t+1} = \max\{0,\ Q_t + c\,(g(x_{t+1}) + \rho)\}.$$

Шаг по x может быть оставлен без изменений (через ∇g); в данном проекте затягивание используется именно как модификация сигнала, поступающего в очередь.

Преимущества.

- Как правило, снижает суммарные нарушения $V(T)$ за счёт более консервативного поведения [3].

Недостатки.

- Может приводить к ухудшению regret, поскольку фактически оптимизация проводится относительно более строгого ограничения.
- Требует подбора ρ: слишком большое значение приводит к существенной деградации качества.

Algorithm 3 DPP-T (DPP with tightening) [3]

Input: $x_t \in X_0$, $Q_t \geq 0$, шаг η, параметры c, ρ
$g_t \leftarrow \nabla f_t(x_t)$
$s_t \leftarrow \nabla g(x_t)$
$x_{t+1} \leftarrow \Pi_{X_0}\big(x_t - \eta(g_t + Q_t s_t)\big)$
$Q_{t+1} \leftarrow \max\{0,\ Q_t + c\,(g(x_{t+1}) + \rho)\}$

3.4 PFS: Online Gradient Descent + Polyak Feasibility Steps

Идея. Метод PFS является основным объектом исследования. Его цель состоит в том, чтобы избежать проекции на истинное множество X, сохранив при этом эффективный контроль допустимости за счёт использования значения ограничения $g(x)$ и его субградиента. Базовая схема включает стандартный шаг онлайн-градиентного спуска в X_0 и дополнительный корректирующий шаг типа Поляка в случае нарушения ограничения [3].

Интуиция шага Поляка. При $g(y) > 0$ требуется уменьшить значение g. Рассмотрим линеаризацию ограничения:

$$g(y - \alpha s) \approx g(y) - \alpha\langle \nabla g(y), s\rangle.$$

При выборе $s = \nabla g(y)$ получаем

$$g(y - \alpha \nabla g(y)) \approx g(y) - \alpha\|\nabla g(y)\|^2.$$

Соответственно, естественный выбор шага имеет порядок

$$\alpha \approx \frac{g(y)}{\|\nabla g(y)\|^2},$$

что соответствует идее шага Поляка; на практике используется стабилизация знаменателя.

Шаг алгоритма.

1. OGD-шаг с проекцией на X_0:

$$y_t = \Pi_{X_0}(x_t - \eta \nabla f_t(x_t)).$$

2. Если $g(y_t) \leq 0$, то положить $x_{t+1} = y_t$.

3. В противном случае выполнить корректирующий feasibility-шаг:

$$x_{t+1} = \Pi_{X_0}\left(y_t - \frac{g(y_t)}{\|\nabla g(y_t)\|^2 + \varepsilon} \nabla g(y_t)\right).$$

Когда применяется.

- В случаях, когда проекция на X затруднена, однако вычисление $g(x)$ и субградиента (или градиента) g является доступным.
- Когда требуется приблизиться по поведению к проекционным методам, избегая вычислительно дорогих проекций [3].

Преимущества.

- Не требует проекции на X и использует только проекцию на X_0.
- Часто обеспечивает малые нарушения за счёт немедленной корректировки при выходе за границу допустимости [3].

Недостатки.

- Корректирующий шаг не является точной проекцией на X, что может приводить к ухудшению regret по сравнению с POGD в случаях, когда Π_X проста.
- Поведение зависит от стабилизатора ε: слишком малое значение может приводить к численной неустойчивости, слишком большое — к недостаточной коррекции.

Algorithm 4 PFS (Polyak Feasibility Steps) [3]

Input: $x_t \in X_0$, шаг η, стабилизатор ε
$g_t \leftarrow \nabla f_t(x_t)$
$y \leftarrow \Pi_{X_0}(x_t - \eta g_t)$
if $g(y) \leq 0$ **then**
    $x_{t+1} \leftarrow y$
else
    $s \leftarrow \nabla g(y)$
    $\alpha \leftarrow \dfrac{g(y)}{\|s\|^2 + \varepsilon}$
    $x_{t+1} \leftarrow \Pi_{X_0}(y - \alpha s)$
end if

4 Реализация и воспроизводимость

Проект (исходный код, конфигурационные файлы и примеры результатов запусков) размещён в открытом репозитории:

`https://github.com/zv3zdochka/oco_pfs_project`.

Реализация выполнена на Python с использованием NumPy, Pandas и Matplotlib. Экспериментальная часть построена таким образом, чтобы вычисления были полностью воспроизводимы на произвольной машине: все параметры задаются в YAML-конфигурациях, а результаты сохраняются в стандартных форматах CSV и PNG.

4.1 Структура проекта

Код организован в виде пакета `oco/` и разделён на функциональные модули:

- `oco/algorithms/` — реализации алгоритмов PFS, DPP, DPP-T и POGD;
- `oco/problems/` — постановки задач (бенчмарки): `toy_quadratic` и `online_logreg`;
- `oco/utils/` — проекции, субградиенты, генерация seed, логирование метрик, а также батч-решатель для вычисления baseline в regret;
- `oco/run_experiment.py` — основной модуль запуска экспериментов;
- `oco/plot_results.py` — построение графиков по сохранённым CSV-файлам.

Конфигурационные файлы экспериментов расположены в директории `configs/`: `configs/toy.ya` и `configs/logreg.yaml`. В них задаются горизонты T, число прогонов (trials) и гиперпараметры алгоритмов.

4.2 Запуск экспериментов

Установка зависимостей выполняется командой:

```
pip install -r requirements.txt
```

Пример запуска экспериментов:

```
python -m oco.run_experiment --config ../configs/toy.yaml
python -m oco.run_experiment --config ../configs/logreg.yaml
```

Построение графиков для ранее сохранённой директории результатов:

```
python -m oco.plot_results --input results/toy/<timestamp>/
python -m oco.plot_results --input results/logreg/<timestamp>/
```

4.3 Сохраняемые артефакты

Каждый запуск создаёт директорию вида `results/<benchmark>/<timestamp>/`, содержащую:

- `metrics_agg.csv` — агрегированные метрики по каждому прогону (в частности, regret, cum_viol, max_viol, cum_loss);

- `metrics_summary.csv` — сводные статистики mean±std по trials для каждого алгоритма и значения T;
- `metrics_step.csv` — пошаговые метрики (для больших горизонтов автоматически используется разрежение по времени);
- `optimal_points.json` — офлайн-оптимальные решения x (используются, в частности, для построения траекторий и sanity-check);
- `config_resolved.yaml` — конфигурация, с которой фактически запускался эксперимент;
- `*.png` — графики (regret, нарушения ограничений, траектории и др.).

4.4 Сопоставимость условий

Для корректного сравнения методы должны наблюдать одну и ту же последовательность данных внутри каждого эксперимента. В реализации используется следующая схема:

1. Для каждой пары (T, trial) заранее генерируются все случайные величины, определяющие поток данных:
   - для Toy-бенчмарка — последовательность $v_1, \ldots, v_T$;
   - для LogReg-бенчмарка — пары $\{(x_t, y_t)\}_{t=1}^{T}$.
2. Далее все алгоритмы прогоняются по одной и той же заранее зафиксированной последовательности.

Начальные seed задаются детерминированно как функция от базового seed, номера прогона и горизонта T, что обеспечивает одновременно воспроизводимость и идентичность данных для всех методов внутри одного trial.

4.5 Метрики и ограничение на число обращений к $g(\cdot)$

Оцениваются две группы метрик:

- качество по потерям (regret):

$$\mathrm{Regret}(T) = \sum_{t=1}^{T} f_t(x_t) - \min_{x \in X} \sum_{t=1}^{T} f_t(x);$$

- качество по соблюдению ограничений — мгновенные и кумулятивные нарушения:

$$v_t = [g(x_t)]_+, \qquad V(T) = \sum_{t=1}^{T} v_t, \qquad \max_t v_t.$$

В реализации отдельно контролируется, что каждый алгоритм выполняет ровно одно обращение к функции ограничения $g(\cdot)$ на шаг. Для этого метод `algo.step()` возвращает пару $(x_t, g(x_t))$, которая далее используется логгером напрямую. Значение потерь $f_t(x_t)$ вычисляется отдельно, так как относится к другому оракулу и не нарушает ограничение “один запрос к ограничению на раунд”.

4.6 Вычисление офлайн-оптимума для regret

Для вычисления regret требуется baseline вида $\min_{x \in X} \sum_t f_t(x)$. В используемых бенчмарках применяется следующий подход:

- **Toy Quadratic:** офлайн-оптимум имеет явную форму. Поскольку $\sum_t 3\|x - v_t\|^2$ минимизируется в точке $x = \frac{1}{T}\sum_t v_t$, далее выполняется проекция среднего на box-ограничение $[-b, b]^d$.
- **Online LogReg:** офлайн-оптимум вычисляется приближённо с помощью батчевого projected gradient descent на шаре $\|w\| \le B$ (используются ранняя остановка и backtracking по шагу).

4.7 Реализованные эксперименты

В проекте реализованы два бенчмарка, запускаемые из отдельных YAML-конфигов:

- **Benchmark A (Toy Quadratic):** $d = 2$, набор горизонтов T (сеткой) и большое число trials; данный бенчмарк удобен для анализа асимптотики regret и метрик нарушений, а также для визуализации двумерных траекторий.
- **Benchmark B (Online LogReg):** $d = 20$, большой горизонт $T = 50000$; данный сценарий отражает постановку онлайн-обучения при больших T, при этом логирование пошаговых метрик автоматически разрежается, чтобы ограничить объём сохраняемых данных.

Таким образом, экспериментальный стенд покрывает два типичных режима: (i) низкоразмерный случай, в котором геометрия ограничений и траектории решений поддаются прямой визуализации, и (ii) высокоразмерную задачу онлайн-обучения при больших горизонтах.

5 Эксперимент 1: синтетическая квадратичная задача Toy Quadratic

5.1 Постановка и офлайн-эталон

Рассматривается задача в размерности $d = 2$. На каждом шаге генерируется вектор $v_t \sim \mathrm{Unif}([0, 1]^2)$, после чего задаётся квадратичная функция потерь (в соответствии с реализацией):

$$f_t(x) = 3\|x - v_t\|_2^2.$$

Функция f_t является выпуклой и L-гладкой, поэтому к ней непосредственно применимы онлайн-градиентные методы [1].

Истинное ограничение задаётся ℓ_∞-box:

$$g(x) = \|x\|_\infty - b, \qquad b = 0.51,$$

и, соответственно, допустимое множество имеет вид

$$X = \{x \in \mathbb{R}^2 : \ |x_i| \le b, \ i = 1, 2\}.$$

В качестве простого множества X_0 используется евклидов шар радиуса $R = 1$: $X_0 = \{x : \|x\|_2 \le 1\}$. Такой выбор обеспечивает включение $X \subset X_0$, тривиальность проекции на X_0, а также простоту проекции на X (покомпонентное отсечение), что делает бенчмарк удобным для проверки корректности реализации.

Regret. Regret вычисляется относительно лучшего фиксированного решения, выбранного апостериори:

$$x^\star \in \arg\min_{x \in X} \sum_{t=1}^{T} f_t(x).$$

Для квадратичной функции потерь $x^\star$ выражается в замкнутой форме: это покомпонентное клиппирование среднего $\bar{v} = \frac{1}{T}\sum_{t=1}^{T} v_t$ на box $[-b, b]^2$ (множитель 3 на структуру решения не влияет). На рисунке с траекториями отмечена точка $x^\star$; визуально наблюдается, что траектории методов концентрируются в окрестности этой точки.

5.2 Параметры эксперимента

Конфигурация эксперимента задаётся файлом `configs/toy.yaml`. Горизонт выбирался из множества $T \in \{2000, 4000, \ldots, 20000\}$, число прогонов (trials) для каждого значения T равно 30. Таким образом, полный запуск соответствует $10 \times 30 \times 4 = 1200$ траекториям, что согласуется с журналом выполнения.

Ключевые гиперпараметры:

$$b = 0.51, \quad R = 1, \quad \varepsilon = 0.25, \quad \rho_{\text{PFS}} = 0.03.$$

Шаги для проекционных методов выбирались в форме $\eta = \eta_{\text{const}}/\sqrt{T}$:

$$\eta_{\text{const}}^{\text{POGD}} = 0.2, \qquad \eta_{\text{const}}^{\text{PFS}} = 0.2, \qquad \eta_{\text{const}}^{\text{DPP-T}} = 0.3.$$

Для DPP и DPP-T использовалась схема с виртуальной очередью и линейной аппроксимацией ограничения, как в примально-двойственных подходах [2]. В варианте DPP-T параметр tightening задавался по схеме, используемой в работе о PFS:

$$\rho = \min\{\varepsilon, \sqrt{c/T}\}, \qquad c = 15,$$

то есть ρ убывает с ростом горизонта [3].

5.3 Результаты

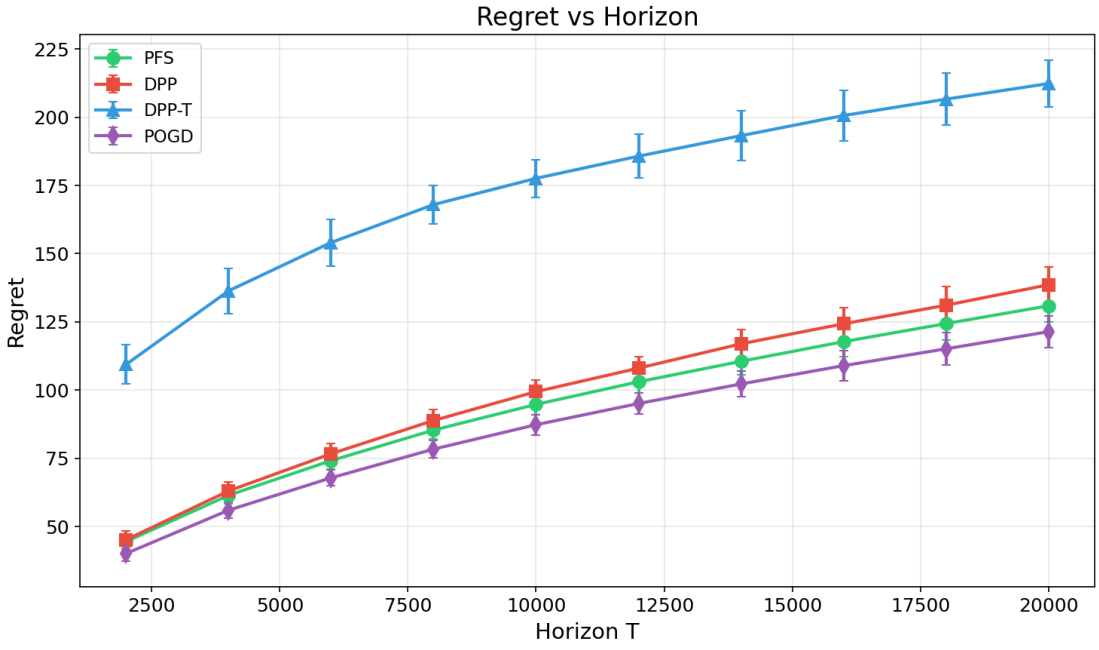

Рис. 1: Toy: regret как функция горизонта T (усреднение по 30 trials).

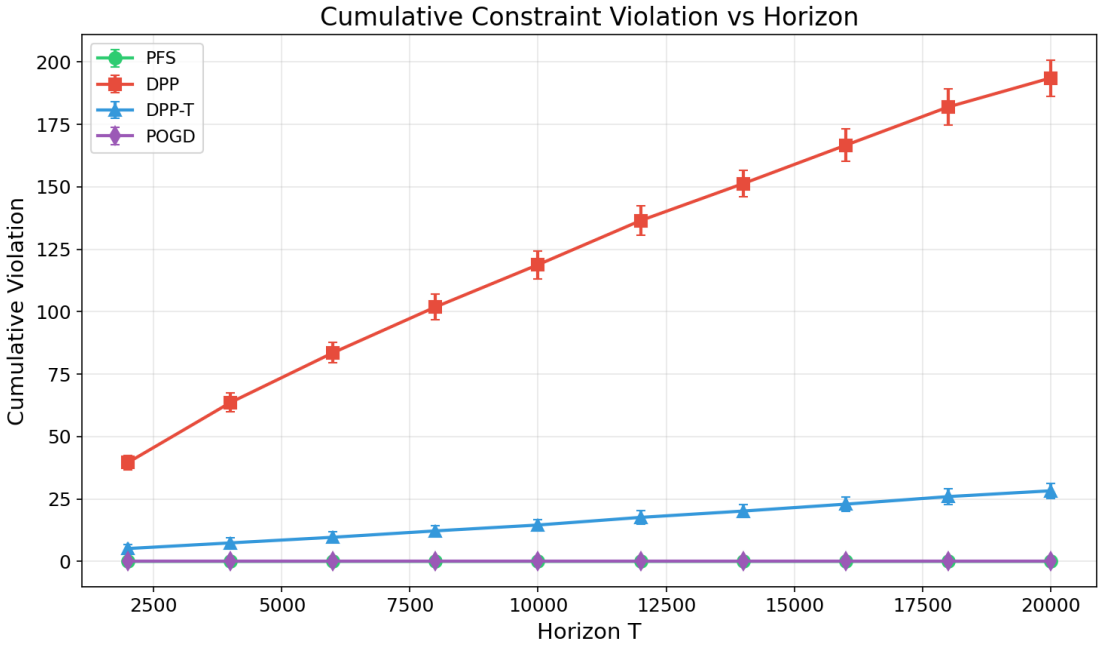

Рис. 2: Toy: кумулятивное нарушение ограничения $V(T)$ в зависимости от горизонта.

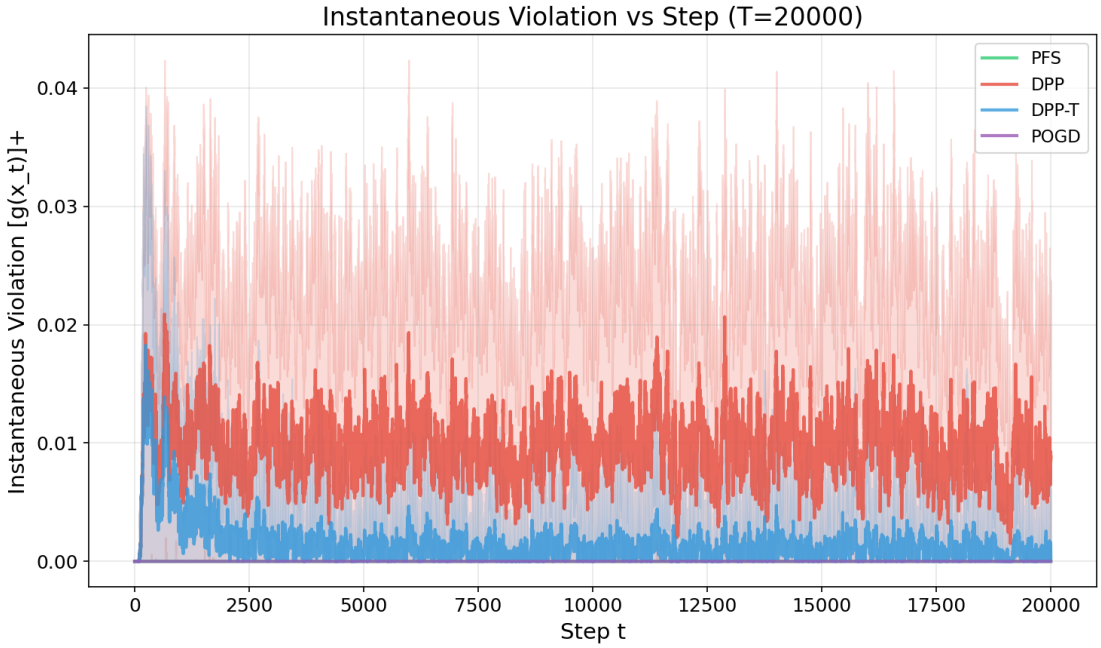

Рис. 3: Toy: мгновенное нарушение $[g(x_t)]_+$ по шагам при $T = 20000$.

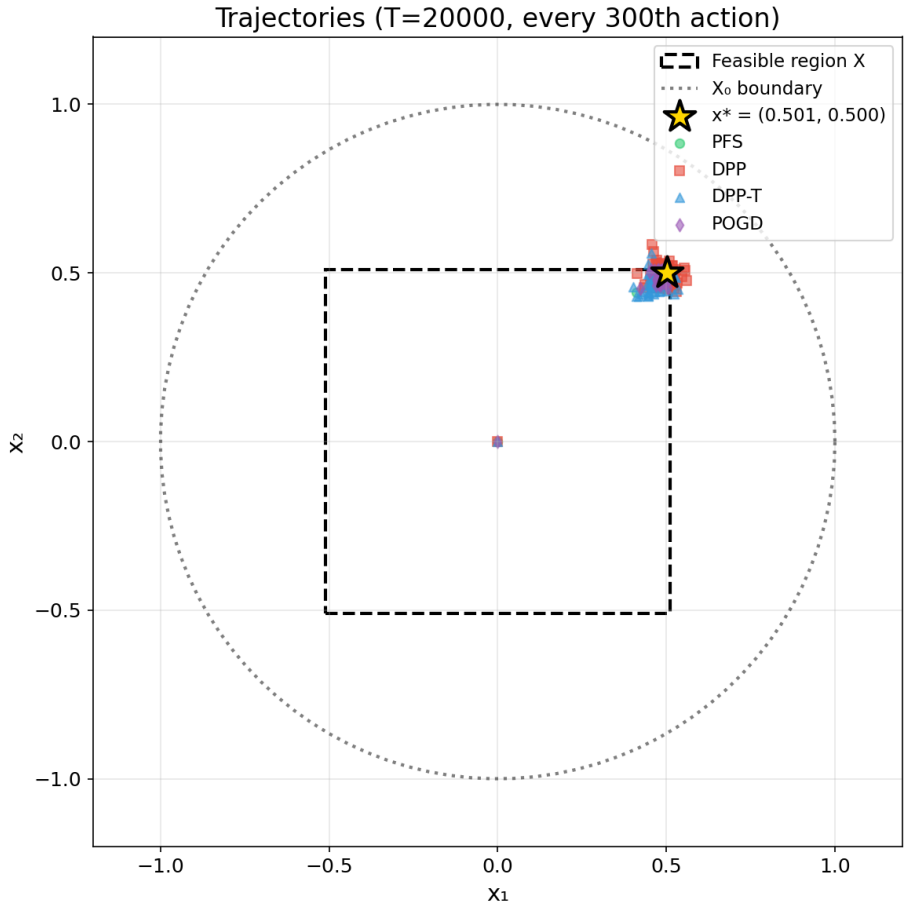

Рис. 4: Toy: траектории в $\mathbb{R}^2$ при $T = 20000$ (показан каждый 300-й шаг), box X и граница шара X_0.

Для фиксации наблюдений в численном виде приведём сводку результатов при $T = 20000$ (среднее $\pm$ std по 30 trials):

Таблица 1: Toy (T=20000): средние значения $\pm$ стандартное отклонение.

Алгоритм	Regret	$V(T)$ (cum. viol)	$\max_t [g(x_t)]_+$
PFS	130.88 ± 3.41	$8.83 \cdot 10^{-5} \pm 5.67 \cdot 10^{-5}$	$6.87 \cdot 10^{-5} \pm 3.28 \cdot 10^{-5}$
DPP	138.59 ± 5.95	193.60 ± 5.53	0.092 ± 0.008
DPP-T	212.35 ± 8.04	28.28 ± 2.86	0.066 ± 0.008
POGD	121.39 ± 4.99	0 ± 0	0 ± 0

5.4 Результаты

Полученные результаты согласуются с ожидаемыми свойствами рассматриваемых подходов [1, 2, 3]:

- POGD обеспечивает строгую допустимость ($V(T) = 0$) и демонстрирует наименьшие значения regret. Данный эффект ожидаем, поскольку проекция на box в рассматриваемой задаче вычислительно проста и удерживает траекторию внутри X.
- PFS также демонстрирует практически нулевые нарушения (значения $V(T)$ порядка 10^{-4}, то есть на уровне численной погрешности) и по regret близок к POGD. На

графике траекторий решения PFS располагаются внутри X и концентрируются в окрестности $x^\star$.

- DPP характеризуется заметными нарушениями: $V(T)$ растёт близко к линейному закону по T, а мгновенные нарушения флуктуируют вокруг положительного уровня. Такое поведение типично для примально-двойственных схем, в которых мгновенная допустимость не гарантируется [2].
- DPP-T снижает нарушения по сравнению с DPP (порядка нескольких раз при $T = 20000$), однако сопровождается существенным ухудшением regret. Тем самым на данном бенчмарке отчётливо проявляется компромисс tightening: уменьшение величины нарушений достигается ценой деградации качества по потерям [3].

В совокупности первый эксперимент служит проверкой корректности реализации: наблюдаемая динамика соответствует проектным свойствам методов (проекционный подход обеспечивает строгую допустимость, примально-двойственные методы допускают контролируемые нарушения, а PFS приближается к проекционному поведению без явной проекции на X).

6 Эксперимент 2: онлайн-логистическая регрессия с ограничением на норму

6.1 Задача

Во втором эксперименте рассматривается типичный сценарий онлайн обучения: на каждом шаге поступает новый пример, а параметры модели обновляются итеративно. Исследуется постановка онлайн-логистической регрессии с ℓ_2-регуляризацией и явным ограничением на норму параметров.

Размерность параметров равна $d = 20$. На шаге t генерируется пример

$$x_t \sim \mathcal{N}(0, I), \qquad y_t = \operatorname{sign}\big((w^\star)^\top x_t + \xi_t\big), \qquad \xi_t \sim \mathcal{N}(0, \sigma^2),$$

где $w^\star \in \mathbb{R}^d$ — фиксированный «истинный» вектор, а ξ_t моделирует аддитивный шум.

Функция потерь задаётся регуляризованной логистической функцией:

$$f_t(w) = \log\big(1 + \exp(-y_t\, w^\top x_t)\big) + \frac{\lambda}{2}\|w\|_2^2, \qquad \lambda = 0.1.$$

Первое слагаемое соответствует ошибке классификации на текущем примере, а ℓ_2-регуляризация стабилизирует значения параметров.

Ограничение задаётся евклидовым шаром:

$$g(w) = \|w\|_2 - B \le 0,$$

то есть требуется $\|w_t\|_2 \le B$ на каждом шаге (для проекционных методов) либо, в случае примально-двойственных методов, контролируется величина нарушений во времени. В качестве простого множества X_0 используется более крупный шар $\|w\|_2 \le R_0$, так что $X \subset X_0$.

Интерпретация ограничения. Ограничение на норму параметров можно интерпретировать как budget/risk constraint: чрезмерно большие веса приводят к нестабильным решениям и потенциально ухудшают обобщающую способность. В терминах constrained OCO это пример ограничения, для которого проекция вычисляется эффективно.

6.2 Настройки

Горизонт фиксирован: $T = 50000$, число прогонов равно 10 (см. `configs/logreg.yaml`). Сравниваются те же четыре алгоритма: PFS, DPP, DPP-T и POGD. Используемые гиперпараметры:

$$\eta_{\text{const}} = 0.5, \quad \varepsilon = 0.1, \quad c = 20, \quad \rho = 0.01, \quad \sigma = 0.3.$$

Как и в первом бенчмарке, шаг обучения в реализации масштабируется как $1/\sqrt{T}$, что обеспечивает сопоставимость настроек при варьировании горизонта.

6.3 Графики по траектории

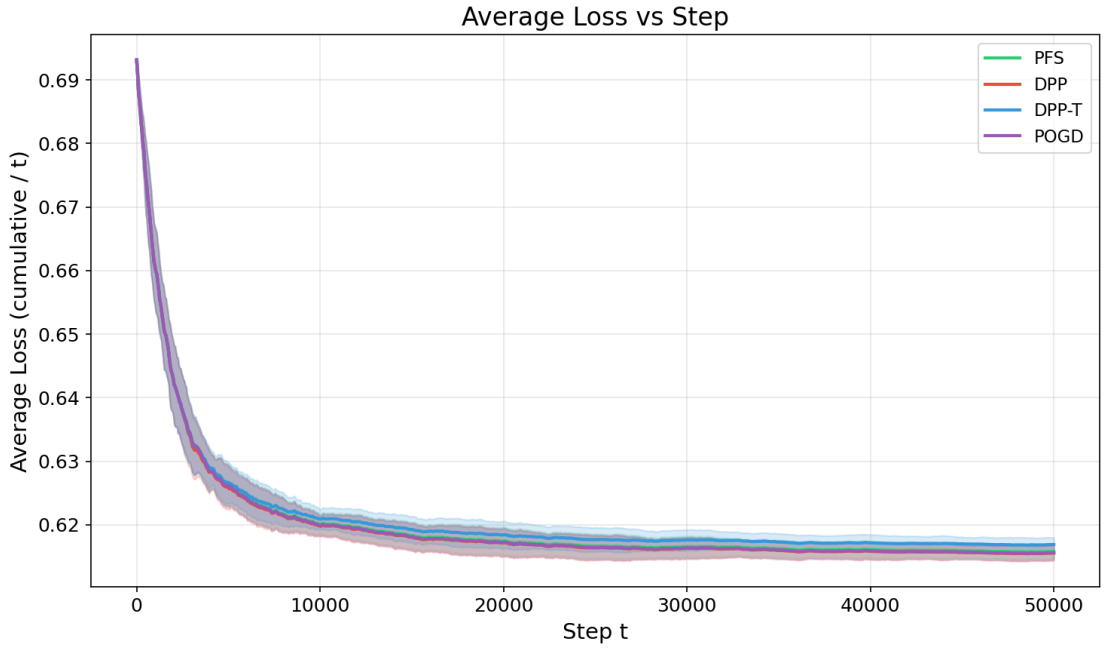

Рис. 5: LogReg: средняя потеря $\frac{1}{t}\sum_{s\le t} f_s(w_s)$ по шагам.

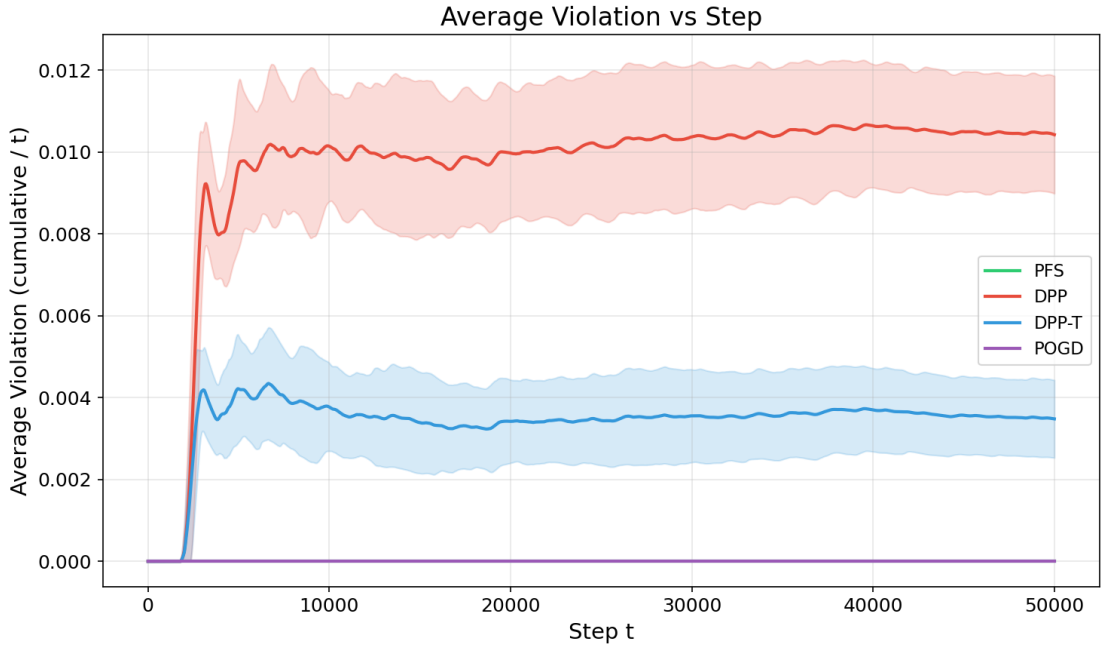

Рис. 6: LogReg: среднее кумулятивное нарушение $\frac{1}{t}\sum_{s\le t}[g(w_s)]_+$ по шагам.

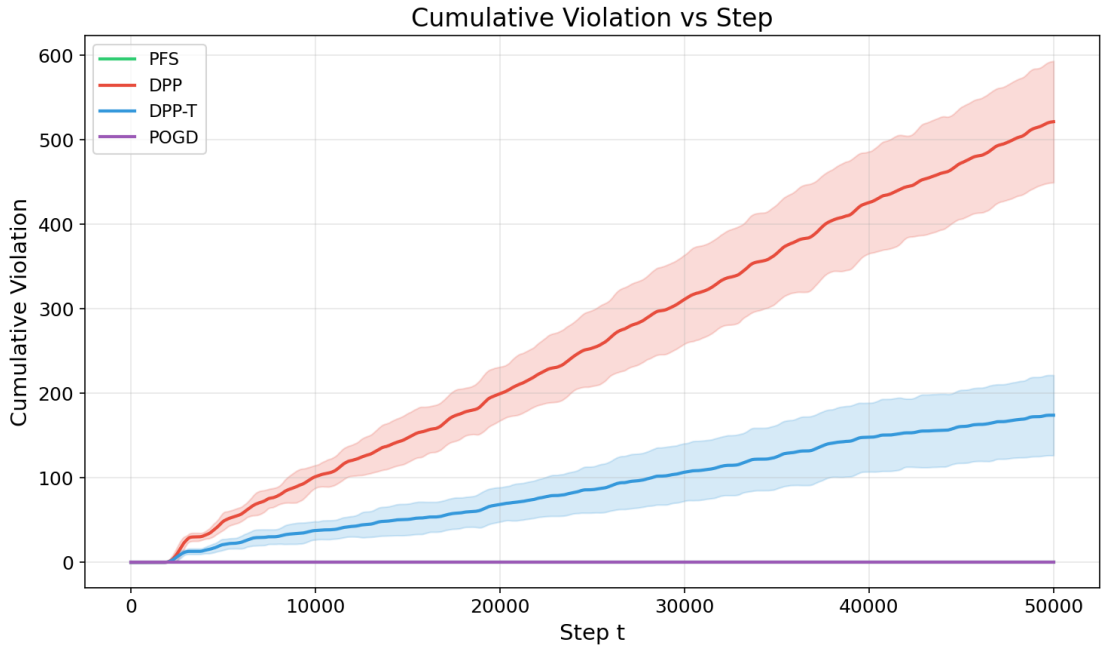

Рис. 7: LogReg: кумулятивное нарушение $V(t) = \sum_{s\le t}[g(w_s)]_+$.

Рис. 8: LogReg: мгновенное нарушение $[g(w_t)]_+$ по шагам.

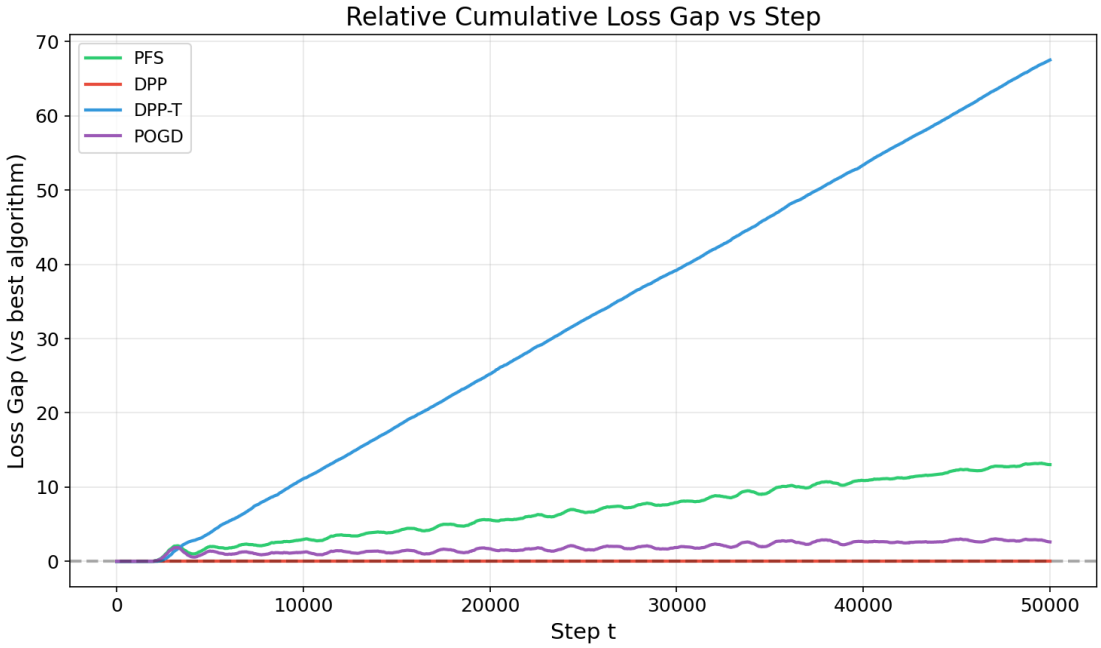

Рис. 9: LogReg: относительный разрыв по кумулятивной потере относительно лучшего алгоритма.

6.4 Итоговые метрики при $T = 50000$

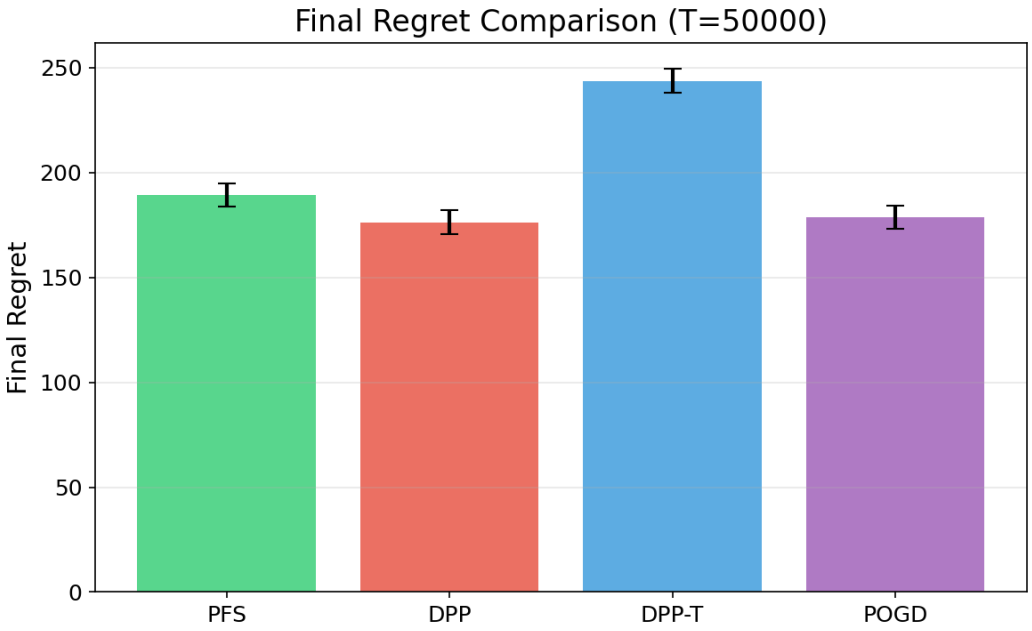

Рис. 10: LogReg: итоговое regret (с доверительными интервалами по trials).

По сводной таблице результатов (усреднение по 10 прогонам):

Таблица 2: LogReg (T=50000): средние значения ± стандартное отклонение.

Алгоритм	Regret	$V(T)$ (cum. viol)	$\max_t[g(w_t)]_+$
PFS	189.27 ± 5.55	0.00 ± 0.00	0.00 ± 0.00
DPP	176.24 ± 5.78	521.37 ± 71.85	0.07 ± 0.01
DPP-T	243.76 ± 5.79	174.14 ± 47.47	0.05 ± 0.01
POGD	178.86 ± 5.56	0.00 ± 0.00	0.00 ± 0.00

Результаты. Эксперимент иллюстрирует типичный компромисс «качество–допустимость» в constrained OCO:

- DPP демонстрирует наименьшее regret, однако сопровождается существенными нарушениями ограничений: кумулятивное нарушение $V(T)$ растёт близко к линейному закону, а среднее нарушение стабилизируется на уровне порядка 10^{-2}.
- DPP-T снижает нарушения относительно DPP (что отражается в меньших значениях $V(T)$ и $\max_t[g(w_t)]_+$), однако regret заметно увеличивается. Данный эффект соответствует интуиции tightening: оптимизация проводится при более жёстком ограничении $g(w) + \rho \leq 0$.
- POGD обеспечивает практически нулевые нарушения и при этом достигает regret, близкий к DPP. Это ожидаемо, поскольку ограничение в данном бенчмарке имеет простую форму (шар), а проекция вычисляется эффективно.
- PFS также обеспечивает нулевые нарушения, однако по regret уступает POGD и DPP. Это согласуется с тем, что PFS избегает точной проекции на X и использует

корректирующий шаг по ограничению, что может приводить к более консервативным обновлениям.

7 Выводы

7.1 Результаты экспериментов

Цель работы носила прикладной характер: реализовать Polyak Feasibility Steps (PFS) и сопоставить его с распространёнными базовыми подходами для constrained OCO — проекционным методом (POGD) и примально-двойственной схемой Drift-Plus-Penalty (DPP) вместе с её модификацией с tightening (DPP-T). Для разделения эффектов, связанных с геометрией допустимого множества и стохастичностью потока данных, использованы два бенчмарка: синтетическая квадратичная задача с простой проекцией и сценарий online learning с шумом.

Эксперимент 1 (Toy Quadratic, box-ограничение). В синтетической задаче с box-ограничением проекция вычисляется эффективно, поэтому POGD выступает естественным ориентиром по regret. Наблюдения подтверждают данное ожидание: POGD обеспечивает наименьший regret на всей сетке горизонтов, а PFS демонстрирует близкие значения. С точки зрения соблюдения ограничений результаты также показательны: POGD и PFS поддерживают около нулевые нарушения на всём горизонте, тогда как DPP приводит к заметному кумулятивному нарушению $V(T)$, растущему с увеличением T. Модификация tightening (DPP-T) уменьшает нарушения по сравнению с DPP, однако сопровождается существенным ухудшением regret. На траекториях в пространстве решений видно, что DPP и DPP-T могут выходить за пределы допустимого box и возвращаться обратно, тогда как PFS и POGD остаются в пределах допустимой области.

Эксперимент 2 (Online Logistic Regression, ограничение на норму). Во втором бенчмарке (онлайн-логистическая регрессия, $T = 50000$) проявляется характерный для constrained OCO компромисс «качество–допустимость». Алгоритм DPP достигает наименьшего regret, но сопровождается заметными нарушениями ограничения.

$$\text{DPP: Regret} = 176.24 \pm 5.78, \quad V(T) = 521.37 \pm 71.85, \quad \max_t [g(w_t)]_+ = 0.07 \pm 0.01.$$

Вариант DPP-T снижает нарушения,

$$\text{DPP-T: Regret} = 243.76 \pm 5.79, \quad V(T) = 174.14 \pm 47.47, \quad \max_t [g(w_t)]_+ = 0.05 \pm 0.01,$$

однако существенно проигрывает по regret.

Алгоритмы PFS и POGD в данном эксперименте обеспечивают практически нулевые нарушения:

$$\text{PFS: } V(T) = 0.00, \quad \max_t [g(w_t)]_+ = 0.00; \qquad \text{POGD: } V(T) \approx 0, \quad \max_t [g(w_t)]_+ \approx 0.$$

Встречающиеся значения порядка 10^{-13} у POGD обусловлены машинной погрешностью вычислений в формате float и по смыслу эквивалентны нулю. По regret POGD представляет собой сильное компромиссное решение:

$$\text{POGD: Regret} = 178.86 \pm 5.56,$$

то есть близко к DPP при отсутствии нарушений. Алгоритм PFS уступает POGD по regret:

$$\text{PFS: Regret} = 189.27 \pm 5.55,$$

что согласуется с тем, что PFS не использует точную проекцию на X, а корректирует недопустимость через feasibility-step Поляка.

7.2 Интерпретация

Наблюдаемая картина объясняется геометрией допустимого множества и вычислительной стоимостью операций проекции и коррекции:

- При простой и вычислительно дешёвой проекции на X метод POGD обеспечивает одновременно строгую допустимость и конкурентоспособные значения regret. Это отчётливо проявляется в Toy-задаче (box) и подтверждается в задаче logreg (шар).
- Алгоритм DPP допускает временные нарушения и компенсирует их через виртуальную очередь; такая стратегия позволяет агрессивнее оптимизировать потери и, как следствие, улучшать regret, однако приводит к накоплению нарушений.
- Алгоритм DPP-T вводит "запас" по ограничению, уменьшая нарушения, но делает обновления более консервативными, что приводит к ухудшению regret (эффект наблюдается в обоих бенчмарках).
- Метод PFS занимает промежуточную позицию: он избегает явной проекции на X, поддерживая допустимость за счёт шага Поляка по ограничению. На рассматриваемых бенчмарках PFS обеспечивает нулевые нарушения, но не превосходит POGD по regret, что согласуется с тем, что проекция в этих постановках дешева.

7.3 Выводы

При наличии эффективной проекции на истинное допустимое множество X метод POGD остаётся наиболее простым и сильным базовым решением: он обеспечивает строгую допустимость и конкурентоспособные значения regret. Алгоритм DPP целесообразен в режимах, где мгновенная допустимость не является критической: он может улучшать качество по потерям, но ценой заметных нарушений ограничения. Вариант DPP-T подтверждает ожидаемый эффект tightening: нарушения уменьшаются, однако regret возрастает.

Метод PFS в экспериментах подтверждает ключевую идею: допустимость можно поддерживать без явной проекции на X, используя значение $g(\cdot)$ и субградиент ограничения. Вместе с тем, в задачах с простой проекцией PFS не даёт выигрыша по regret относительно POGD. Практически это означает, что наибольшая ценность PFS проявляется в постановках, где проекция на X сложна или дорога, тогда как вычисление $g(x)$ и $\nabla g(x)$ остаётся доступным по цене; в таких режимах PFS может выступать реальной альтернативой проекционным методам [3].

7.4 Возможные направления развития

Дальнейшее развитие работы можно усилить следующими направлениями:

- Рассмотреть режим, в котором проекция на X действительно вычислительно затратна (например, X задано пересечением нескольких ограничений или не допускает простой формулы проекции). В этом случае целесообразно сравнивать не только regret/violation, но и вычислительное время шага.
- Провести анализ чувствительности по ключевым гиперпараметрам (η, c, ρ, ε) и продемонстрировать устойчивость выводов.
- Рассмотреть несколько ограничений $g_i(x) \leq 0$ и оценить поведение алгоритмов в более общей постановке constrained OCO.

Список литературы

[1] E. Hazan. *Introduction to Online Convex Optimization*. arXiv preprint arXiv:1909.05207, 2019.

[2] H. Yu, M. J. Neely. *Online Convex Optimization with Stochastic Constraints*. In: Proceedings of the 30th Conference on Neural Information Processing Systems (NeurIPS), 2017. Также доступно: arXiv:1708.04726.

[3] *Constrained Online Convex Optimization with Polyak Feasibility Steps*. arXiv preprint arXiv:2502.13112, 2025.

[4] *Adaptive Methods for Online Convex Optimization*. arXiv preprint arXiv:2002.03963, 2020.

[5] Яндекс Образование. *Онлайн-обучение и стохастическая оптимизация*. Доступно по адресу: `https://education.yandex.ru/handbook/ml/article/onlajn-obuchenie-i-stohasticheskaya-optimizaciya`