Санкт-Петербургский Государственный Университет
Математико-механический факультет

Программная инженерия

Турсунова Мунира Бахромовна

# Система для моделирования алгоритмов стохастической оптимизации для задач трекинга

Выпускная квалификационная работа бакалавра

Научный руководитель:
проф. кафедры СП, д. ф.-м. н. О. Н. Граничин

Рецензент:
ст. н. с. ИПМаш РАН, к. ф.-м. н. Ю. В. Иванский

Санкт-Петербург
2020

SAINT PETERSBURG STATE UNIVERSITY

Software Engineering

Munira Tursunova

# System for modeling stochastic optimization algorithms for tracking

Bachelor's Thesis

Scientific supervisor:
Professor, Dr. of Sci. Oleg Granichin

Reviewer:
Senior researcher IPME RAS, Ph.D. Yury Iwanskiy

Saint Petersburg
2020

# Оглавление

# Введение

Сегодня мы можем наблюдать всестороннее развитие различных областей науки с применением систем искусственного интеллекта. Обычному человеку, ещё полвека назад, было бы сложно вообразить, что компьютеры все активнее будут замещать человеческие ресурсы и отодвигать на второй план многие специальности. Так например, автоматизация работ крупных промышленных предприятий сокращает количество сотрудников основного производства: достаточно нанять квалифицированный штат программистов, осуществляющих те или иные алгоритмы. Хранение огромных баз данных больше не занимает много места, а играть в шахматы не обязательно с напарником. Человечество научилось обучать различные сети подобно человеческому мозгу, что привело к технической революции: мы пользуемся различными приложениями, гаджеты нам прокладывают маршрут, активно работают системы распознавания лиц, голоса, и, даже, можно с помощью компьютера получить медицинский диагноз и соответсвующие рекомендации. Список задач, которые способен решить сегодня искусственный интеллект невообразим и, благодаря всестороннему развитию, он расширяется экспоненциально. Как правило, в основе всего вышеперечисленного лежит знание класса методов искусственного интеллекта известное как «Машинное обучение». Алгоритмы машинного обучения широко используются в самых различных областях, таких как программирование или же, например, медицина. Методы машинного обучения можно применить в распознавании речи и образов, в медицинской и технической диагностике, в биоинформатике и во многом другом. Довольно частый случай применения методов машинного обучения — это так называемые задачи прогнозирования. В задачах прогнозирования обычно наиболее эффективным и распространенным методом решения является регрессионный анализ, в основе которого лежат градиентные методы.

Градиентные методы берут свое начало еще в далеком 1847 году, когда Коши представил общественности метод градиентного спуска.

С тех пор появилось множество методов численной оптимизации, которые в большинстве своём были модификацией градиентного спуска [23, 18, 29, 12, 25, 15, 13, 6, 2, 22]. Тем не менее, на сегодняшний день большей популярностью пользуются ускоренные градиентные методы вследствие быстрой скорости сходимости. Одним из первых таких методов является метод тяжелого шарика, выдвинутый Б. Поляком в середине прошлого века [23], однако сейчас метод тяжелого шарика редко используется на практике, а вместо него используется быстрый градиентный метод Нестерова, являющийся модификацией метода тяжелого шарика [18, 19]. Метод Нестерова показывает отличные результаты в задачах стационарной детерминированной оптимизации. Однако в действительности, в жизни мы намного чаще имеем дело с данными, которые в связи с неидеальностью измерений носят не детерминированный характер, а в большинстве своем искажены посторонним шумом. Это и привело к тому, что с развитием технологий начали появляться стохастические методы, которые показали хорошие результаты [23, 18, 12, 25, 2, 27, 13, 14, 15, 24]. В частности, только за последние несколько десятилетий появилось огромное множество работ, посвященных стохастической оптимизации [19, 29, 13, 28, 3]. Приложения стохастической оптимизации широко применяются на практике; к примеру, они встречаются в реальных системах, таких как адаптивная обработка сигналов, адаптивное распределение ресурсов в сетевых системах, идентификация систем и адаптивное управление [20, 7, 31, 5]. Обычно решение стохастической задачи оптимизации в условиях неопределенности сводится к поиску набора системных параметров, которые обеспечивают минимальное или максимальное значение для определенной функции потерь. На практике эти параметры также часто могут изменяться со временем. В этом случае распределение данных становится нестационарным [25, 29, 6, 11, 4]. Примером нестационарной оптимизации для нелинейной функции является онлайн обучение глубокой нейронной сети.

Другой проблемой помимо нестационарности и стохастичности данных является то, что бывают случаи, когда вычисление градиента функ-

ции ведет к большим вычислительным затратам или же когда градиент функции попросту неизвестен. В таких случаях было бы полезно уметь решать задачу нестационарной стохастической оптимизации, имея только значение функций.

На сегодняшний день насчитывается немалое количество алгоритмов стохастической нестационарной оптимизации, и не всегда понятно, какой метод подойдёт лучше для решения той или иной задачи. Это указывает на необходимость такого инструмента, с помощью которого можно было бы сравнивать различные алгоритмы оптимизации.

Начиная со второй половины прошлого века и по сей день было создано немало систем для решения разнообразных задач оптимизации "— линейных, нелинейных, дискретных, квадратичных, смешанно-целочисленной и многих других [1, 32, 21, 17, 26, 8, 16, 30, 9]. Тем не менее, до сих пор не существует системы, в которой можно было бы задавать и модель функции, и модель шума, и модель дрифта, настраивать такие параметры, как количество итераций и количество шагов в модели, а также добавлять новые пользовательские алгоритмы в систему. Более того, все существующие системы математической оптимизации решают задачу нестационарной оптимизации, но не предназначены для задачи отслеживания параметров, в которой данные нестационарные, иными словами, изменяются со временем. Возможность задавать не только модель функции, но и модель шума и дрифта позволяет решать более широкий класс задач, а возможность добавления новых алгоритмов в систему позволяет адаптировать систему для решения таких задач оптимизации, для которых предоставленных алгоритмов недостаточно. Также возможность добавления новых алгоритмов и возможность настройки таких параметров, как количество итераций и шагов, позволяет тестировать и сравнивать алгоритмы, что облегчает процесс создания новых алгоритмов нестационарной стохастической оптимизации.

Всё вышеизложенное указывает на актуальность создания системы для моделирования различных алгоритмов стохастической оптимизации для задач трекинга, с возможностью задания модели функции, мо-

дели шума и модели дрифта, а также на актуальность создания нового алгоритма нестационарной стохастической оптимизации, не использующего знаний о значениях градиента функции, а пользующийся только измерениями самой функции.

# 1. Постановка задачи

Целью работы является разработка прототипа системы для моделирования различных алгоритмов стохастической оптимизации для задач трекинга, с возможностью задания модели функции, модели шума и модели дрифта, а также создание и реализация нового алгоритма нестационарной стохастической оптимизации «Рандомизированный быстрый квази-градиентный метод для задач трекинга». Для достижения вышеупомянутого результата были выдвинуты следующие задачи:

- Исследовать существующие алгоритмы оптимизации, применимые в задаче трекинга.
- Разработать новый алгоритм нестационарной стохастической оптимизации «Рандомизированный быстрый квази-градиентный метод для задач трекинга» и исследовать его свойства.
- Исследовать существующие системы для моделирования алгоритмов оптимизации.
- Составить требования к системе.
- Разработать архитектуру системы.
- Реализовать классические методы стохастической оптимизации и новый метод «Рандомизированный быстрый квази-градиентный метод для задач трекинга» в системе.
- Разработать прототип системы, включающий в себя возможность визуализации невязок алгоритмов.
- Сделать графический интерфейс в системе.
- Провести апробацию системы.

# 2. Алгоритмы нестационарной стохастической оптимизации

## 2.1. Постановка задачи нестационарной стохастической оптимизации

Главной задачей алгоритмов оптимизации является поиск такого параметра $\theta \in \mathbb{R}^q$, где $q$ — размерность пространства, минимизирующий соответствующую дифференцируемую функцию потерь $f : \Theta \to \mathbb{R}$, т.е.:

$$\text{Найти} \quad \theta = \text{Argmin}_{\theta \in \Theta} f(\theta). \tag{1}$$

В задачах трекинга же, данные меняются с течением времени, следовательно и функции потерь $f_n$, описывающие эти данные, также меняются со временем. В следствие чего алгоритмы оптимизации для задач трекинга позволяют решать задачу поиска последовательности параметров $\theta_n \in \mathbb{R}^q$, которые минимизируют соответствующие дифференцируемые функции потерь $f_n$, т.е.:

$$\text{Найти} \quad \theta_n = \text{Argmin}_{\theta \in \Theta} f_n(\theta), \quad \forall n = 0, 1, 2, \ldots. \tag{2}$$

Здесь и далее $n$ — момент времени.

(2) также называют задачей отслеживания параметров.

В теории обычно предполагается, что $f$ — дифференцируемые и выпуклые функции и множество $\Theta$, на котором они определены, является выпуклым.

Градиентные методы основываются на следующем утверждении:

$$\text{Пусть } \theta_n^* \text{ - точка минимума, тогда } \nabla f_n(\theta_n^*) = 0.$$

Допустим, что дана точка $\theta_n$ и надо найти такую $\hat{\theta}_n$, что $f_n(\hat{\theta}_n) < f_n(\theta_n)$. Для этого запишем

$$f_n(\hat{\theta}_n) \approx f(\theta_n) + \nabla f_n(\theta_n)(\hat{\theta}_n - \theta_n) \tag{3}$$

Пусть $h > 0$ и значение новой точки $\hat{\theta}_n$ вычисляется по формуле:

$$\hat{\theta}_n = \theta_n - h\|\nabla f_n(\theta_n)\| \tag{4}$$

Здесь и далее $h$ — скорость обучения.

Тогда подставив (4) в (3), получаем:

$$f_n(\hat{\theta}_n) \approx f(\theta_n) + \nabla f_n(\theta_n)(\theta_n - h\|\nabla f_n(\theta_n)\| - \theta_n) =$$
$$= f(\theta_n) - h\|\nabla f_n(\theta_n)\|^2 < f(\theta_n)$$

## 2.2. Стохастический градиентный спуск

Учитывая всё вышеизложенное, становится возможным найти по точке $\theta_n$ такую $\hat{\theta}_n$, что $f_n(\hat{\theta}_n) < f_n(\theta_n)$. Теперь, чтобы найти минимум функции, нужно повторить эту операцию некоторое число раз и в итоге получим метод градиентного спуска:

$$\hat{\theta}_{n+1} = \hat{\theta}_n - h\|\nabla f_n(\hat{\theta}_n)\| \tag{5}$$

Однако же в (5) рассматривается детерминированная модель, где измерения функций получены без шума. В случае стохастических алгоритмов нестационарной оптимизации измерения градиента $Y_n(\hat{\theta}_n)$ искажены аддитивным шумом $\xi_n \in \mathbb{R}^q$:

$$\begin{gathered} Y_n(\hat{\theta}_n) = \nabla f_n(\hat{\theta}_n) + \xi_n, \\ \mathbb{E}Y_n(\hat{\theta}_n) = \nabla f_n(\hat{\theta}_n) \end{gathered} \tag{6}$$

И, соответственно, стохастический градиентный спуск имеет следующую формулу:

$$\hat{\theta}_{n+1} = \hat{\theta}_n - h\|Y_n(\hat{\theta}_n)\| \tag{7}$$

где $\mathbb{E}$ - символ математического ожидания Важно также помнить, что скорость обучения $h < \frac{2}{L}$, где $L$ — общая константа Липшица функций $f_n$, гарантирует убывание функций.

## 2.3. Быстрый градиентный метод Нестерова

В быстром градиентном методе Нестерова алгоритм также как и в (7) позволяет уменьшать значения функций двигаясь против градиента, но при этом он также двигается в том же направлении, что и на предыдущей итерации. Другими словами, общую формулу итеративного поиска оценки точки минимума можно записать следующим образом:

$$\hat{\theta}_{n+1} = \hat{\theta}_n - hY(\hat{\theta}_n + \alpha_n(\hat{\theta}_n - \hat{\theta}_{n-1})) + \alpha_n(\hat{\theta}_n - \hat{\theta}_{n-1})$$

Параметр $\alpha_n$ также вычисляется итеративно.

Алгоритм быстрого градиента Нестерова представляет собой последовательность следующих шагов:

1. Выбираем шаг $h > 0$
2. Начинаем с начальной оценки $\hat{\theta}_0 \in \mathbb{R}^q$
   Выбираем $\gamma_0 > 0$, $v_0 = \hat{\theta}_0$
3. На $n$-ой итерации ($n \geq 0$):
   (a) Найти $\alpha_n \in (0, 1)$ т. ч.:
   $$\alpha_n^2 = (1 - \alpha_n)\gamma_n h + \alpha_n \mu h.$$
   (b) Присвоить $\gamma_{n+1} = (1 - \alpha_n)\gamma_n + \alpha_n \mu$.
   (c) Вычислить $x_n = \frac{1}{\gamma_n + \alpha_n \mu}\Big(\alpha_n \gamma_n v_n + \gamma_{n+1}\hat{\theta}_n\Big)$
   (d) Вычислить значение $Y_n(x_n)$.
   (e) Найти новую оценку $\hat{\theta}_{n+1}$: $\hat{\theta}_{n+1} = x_n - hY_n(x_n)$.
   (f) Вычислить $v_{n+1} = \frac{1}{\gamma_{n+1}}\Big[(1 - \alpha_n)\gamma_n v_n + \alpha_n \mu x_n - \alpha_n Y_n(x_n)\Big]$.

## 2.4. Быстрый стохастический градиентный метод для задач трекинга

Метод быстрого стохастического градиента для задач трекинга (англ. Stochastic fast gradient for traking (SFGT)) также позволяет решать задачу (1). В нем рассматривается модель, в которой единственной доступной информацией служат измерения градиента $Y_n(\theta)$, искажённые аддитивным шумом $\xi_n \in \mathbb{R}^q$:

$$Y_n(\theta) = \nabla f_n(\theta) + \xi_n, \quad n \in \mathbb{N}. \tag{8}$$

В алгоритме также вводятся дополнительные ограничения на скорость изменения функций и градиента функций, помимо общих ограничений на выпуклость и Липшицевость функций.

Алгоритм SFGT заключается в следующем:

1. Выбираются значения параметров $h > 0$, $\eta \in (0, \mu)$, $\alpha_x \in (0, 1)$ так, что значение $\alpha_n$, удовлетворяющее неравенству (9), всегда может быть найдено.

2. Выбираются значения $\hat{\theta}_0 \in \mathbb{R}^q$, $\gamma_0 > 0$.
   Фиксируется значение $v_0 = \hat{\theta}_0$.
   Вычисляется значение переменной $H_1$: $H_1 = h - \frac{h^2 L}{2}$.

3. На $n$-ой итерации ($n \geq 0$):

   (a) Найти $\alpha_n \in [\alpha_x, 1)$ такое, что

   $$H_1 - \frac{\alpha_n^2}{2\gamma_{n+1}} > 0. \tag{9}$$

   (b) Вычислить $\gamma_{n+1} = (1 - \alpha_n)\gamma_n + \alpha_n(\mu - \eta)$.

   (c) Вычислить $x_n = \frac{1}{\gamma_n + \alpha_n(\mu - \eta)}\Big(\alpha_n \gamma_n v_n + \gamma_{n+1}\hat{\theta}_n\Big)$

   (d) Рассчитать значение $Y_n(x_n)$.

   (e) Найти новую оценку $\hat{\theta}_{n+1}$: $\hat{\theta}_n = x_n - hY_n(x_n)$.

   (f) Установить $v_{n+1} = \frac{1}{\gamma_{n+1}}\Big[(1 - \alpha_n)\gamma_n v_n + \alpha_n(\mu - \eta)x_n - \alpha_n Y_n(x_n)\Big]$.

## 2.5. Рандомизированный быстрый квази-градиентный метод для задач трекинга

### 2.5.1. Постановка задачи и модель

Новый алгоритм «Рандомизированный быстрый квази-градиентный метод для задач трекинга» (RFQGT) позволяет решить задачу поиска последовательности параметров $\theta_n \in \mathbb{R}^q$, которые минимизируют соответствующие дифференцируемые функции потерь, иными словами позволяет решить задачу отслеживания параметров (2).

Рассматривается модель, в которой единственной доступной информацией служат измерения $y_n(\theta)$, искаженные аддитивным шумом $\xi_n \in \mathbb{R}^q$:

$$y_n(\theta) = f_n(\theta) + \xi_n, \quad n = 0, 1, 2, \ldots. \tag{10}$$

Пусть $(\Omega, \mathcal{F}, P)$ - базовое вероятностное пространство, соответствующее выборочному пространству $\Omega$ с $\sigma$ -алгеброй всех событий $\mathcal{F}$, вероятностной мерой $P$.

Обозначим $\mathcal{F}_n$ $\sigma$-алгебру всех вероятностных событий, произошедщих до момента времени $n = 1, 2, \ldots$.

$\mathbb{E}_{\mathcal{F}_n}$ — символ условного математического ожидания относительно $\sigma$-algebra $\mathcal{F}_n$.

Алгоритм RFQGT, предлагаемый в этой работе, находит последовательность таких оценок $\{\hat{\theta}_n\}_{n=0}^{\infty}$, которые являются решением следующей задачи:

$$\begin{gathered} \text{Найти} \quad \hat{\theta}_n \quad \text{т.ч.} \quad \exists N, C < \infty : \quad \forall n > N \\ \mathbb{E}\|\hat{\theta}_n - \theta_t\|^2 \leq C. \end{gathered} \tag{11}$$

Обозначим $f_n^* = f_n(\theta_t)$.

Предполагаем, что функции $f_n$ и шум $\xi_n$ удовлетворяют следующим свойствам:

*Предположение 1.* Функции $f_n$ имеют общую константу Липшица

$L > 0$ и общий параметр сильной выпуклости $\mu > 0$:

$$\begin{gathered} \forall x \in \mathbb{R}^q \quad \|\nabla f_n(x)\| \leq L\|x - \theta_n\|, \\ \langle \nabla f_t(x), x - \theta_n \rangle \geq \mu \|x - \theta_n\|^2. \end{gathered} \tag{12}$$

*Предположение 2.* Для любого $n \geq 0$, изменение функции ограничено:

$$\|f_{2n}(x) - f_{2n+2}(x)\| \leq a\|\nabla f_n(x)\| + b, \quad f_n^* = f^* \tag{13}$$

$$\|\nabla f_{2n+2}(x) - \nabla f_{2n}(x)\| \leq c. \tag{14}$$

*Предположение 3.* Для $n = 1, 2, \ldots$, последовательности разностей $\tilde{\xi}_n = \xi_{2n} - \xi_{2n-1}$ наблюдаемого шума ограничены: $|\tilde{\xi}_n| \leq \sigma < \infty$, $\mathbb{E}\tilde{\xi}_n^2 \leq \sigma^2$ если последовательность $\{\tilde{\xi}_n\}$ случайная.

Пусть $\Delta_n$ $n = 1, 2, \ldots$ — наблюдаемые последовательности независимых случайных векторов из $\mathbb{R}^q$. $\Delta_n$ имеют распределение Бернулли, причем каждый компонент независимо принимает значения $\pm\frac{1}{\sqrt{q}}$ с вероятностями $\frac{1}{2}$. Эта последовательность обычно называется *simultaneous test perturbation.*

*Предположение 4.*

a) Вектора $\Delta_n$, $n = 1, 2, \ldots$, независимы;

b) Вектора $\Delta_n$ и $\theta_{2n-1}, \theta_{2n}$ (если они случайные) не зависят от $\sigma$-алгебры $\mathcal{F}_{2k-2}$;

c) Если $\theta_{2n-1}, \theta_{2n}, \tilde{\xi}_n$ являются случайными, то случайные векторы $\Delta_n$ и элементы $\theta_{2n-1}, \theta_{2n}, \tilde{\xi}_n$ являются независимыми.

### 2.5.2. Алгоритм

Для решения задачи, определенной в (11) с моделью наблюдения, определенной в (10), удовлетворяющей предположениям 1–4 был разработан RFQGM алгоритм, представляющий последовательность следующих шагов:

1. Выбираются значения параметров $h > 0$, $\beta > 0$, $\eta \in (0, \mu)$, $\alpha_x \in (0, 1)$, так что значение $\alpha_n$, удовлетворяющее неравенству (15), всегда может быть найдено.

2. Выбираются значения $\hat{\theta}_0 \in \mathbb{R}^q$, $\gamma_0 > 0$. Фиксируется значение $v_0 = \hat{\theta}_0$. Вычисляется значение переменной $H$: $H = h - \frac{h^2 L}{2}$.

3. На $n$-ой итерации ($n := n + 1$):

   (a) Найти $\alpha_n \in [\alpha_x, 1)$ такое, что

   $$H - \frac{\alpha_n^2}{2\gamma_{n+1}} > 0. \tag{15}$$

   (b) Вычислить $\gamma_n = (1 - \alpha_n)\gamma_{n-1} + \alpha_n(\mu - \eta)$.

   (c) Вычислить

   $$x_n = \frac{1}{\gamma_n + \alpha_n\gamma_{n-1}}\Big(\alpha_n\gamma_{n-1}v_{n-1} + \gamma_n\hat{\theta}_{2n}\Big).$$

   (d) Сгенерировать $\Delta_n$, используя распределение Бернулли, где каждый элемент независимо принимает значения $\pm\frac{1}{\sqrt{q}}$ с вероятностями $\frac{1}{2}$.

   (e) Вычислить значения двух новых наблюдений:

   $$y_n^{\pm} = y_{2n-1+\frac{1}{2}(1\pm1)}(x_n \pm \beta\Delta_n).$$

   (f) Найти новые оценки: $\hat{\theta}_{n+1}$:

   $$\hat{\theta}_{2n-1} = \hat{\theta}_{2(n-1)},$$

   $$\hat{\theta}_{2(n)} = x_n - h\bar{Y}_n,\ \bar{Y}_n = \Delta_n\frac{y_n^+ - y_n^-}{2\beta}.$$

   (g) Установить

   $$v_{n+1} = \gamma_{n+1}^{-1}((1 - \alpha_n)\gamma_n v_n + \alpha_n(\mu - \eta)x_n - \alpha_n\bar{Y}_n(x_n) + \alpha_n L\beta).$$

**Замечание 1.** Набор параметров, удовлетворяющий всем условиям, существует. Можно выбрать $0 \le \eta \le \mu$, $\alpha_x < \sqrt{\frac{\mu-\eta}{L}}$ и $\gamma_0 > \mu - \eta$, $h = L^{-1}$. Тогда получим

$$\gamma_n > \mu - \eta$$

Доказательство по индукции. Из условия выбора параметров имеем:

$$\gamma_0 > \mu - \eta$$

Пусть $\gamma_n > \mu - \eta$, тогда:

$$\gamma_{n+1} = (1 - \alpha_n)\gamma_n + \alpha_n(\mu - \eta) > \mu - \eta$$

В результате имеем, что выбор $\alpha_x$ гарантирует, что неравенство (15) выполняется всегда для $\alpha_x$:

Из условия выбора параметров имеем:

$$\alpha_x < \sqrt{\frac{\mu - \eta}{L}}$$

Перепишем (15) в следующем виде:

$$\alpha_n < \sqrt{2H\gamma_{n+1}} = \sqrt{\frac{\gamma_{n+1}}{L}}$$

Тогда имеем:

$$\sqrt{\gamma_{n+1}L^{-1}} > \sqrt{(\mu - \eta)L^{-1}} > \alpha_x$$

### 2.5.3. Свойства алгоритма

Для доказательства сходимости метода введем следующие обозначения:

Пусть $\{\alpha_n\}_{n=0}^{\infty}$, $\{\lambda_n\}_{n=0}^{\infty}$, $\{A_n\}_{n=0}^{\infty}$, $\{Z_n\}_{n=0}^{\infty}$ и $\{D_n\}_{n=0}^{\infty}$ последовательности определенные следующим образом:

$$\alpha_n \in [\alpha_x, 1), \quad \lambda_0 = 1, \quad \lambda_{n+1} = (1 - \alpha_n)\lambda_n \tag{16}$$

$$\begin{gathered} A_0 = 0, \\ A_{n+1} = (1 - \alpha_n)((1 - \lambda_n)a + A_n), \\ Z_n = (1 - \lambda)(b + ac) + A_n c, \end{gathered} \tag{17}$$

$$
\begin{gathered}
D_0 = 0, \\
D_{n+1} = (1-\alpha_n)D_n + \frac{a(1+\alpha_n)+hc}{4\epsilon} + (1+\alpha_n)b + \\
+(1-\alpha_n)Z_n + h^2\frac{L}{2}\sigma^2 + \frac{\alpha_n c^2}{2\eta}.
\end{gathered} \tag{18}
$$

Тогда имеем

$$
D_\infty = \alpha_x^{-1}\Big[\frac{2a+hc}{4\epsilon} + 2b + (1-\alpha_x)(b + A_\infty c) + h^2\frac{L}{2}\sigma^2 + \frac{c^2}{2\eta}\Big] \tag{19}
$$

где

$$
\begin{gathered}
\Gamma = \max_{n\geq 0}\gamma_n, \\
\epsilon \in \left(0, \frac{1}{a(1+\alpha_x)+hc}\left(H_1 - \frac{\alpha_x^2}{2\Gamma}\right)\right]
\end{gathered} \tag{20}
$$

**Теорема 1** *Если предположения 1–4 выполнены, то RFQGM, описанный выше, решает проблему (11) со следующими параметрами*

$$
C = \frac{2}{\mu}D_\infty, \tag{21}
$$

*для $\alpha_x$, $\eta$, $h$ выбранных в алгоритме.*

*Ошибка оценки после конечного числа итераций ограничена как:*

$$
\mathbb{E}_n f_n(\hat{\theta}_n) - f_n^* \leq \prod_{i=1}^{n}(1-\alpha_n)(\phi_0(\theta_0) - f^* + \Phi) + D_n, \tag{22}
$$

*где*

$$
\begin{gathered}
\phi_0(x) = f_0(\hat{\theta}_0) + \frac{\gamma_0}{2}\|x - v_0\|^2, \\
\Phi = \frac{\gamma_0 c^2}{2\mu^2}
\end{gathered} \tag{23}
$$

**Замечание 2** Из предположения 1 получим следующее неравенство:

$$
\begin{gathered}
\|\hat{\theta}_n - \theta_n\|^2 \leq \mu^{-1}\left\langle \nabla f_n(\hat{\theta}_n), \hat{\theta}_n - \theta_n \right\rangle \approx \mu^{-1}(f_n(\hat{\theta}_n) - f_n(\theta_n)) \\
\mathbb{E}\|\hat{\theta}_n - \theta_n\|^2 \leq \mu^{-1}(\mathbb{E}f_n(\hat{\theta}_n) - f_n(\theta_n))
\end{gathered} \tag{24}
$$

### 2.5.4. Доказательство

Доказательство теоремы 1 во многом схоже с доказательством в статье [29], за исключением модели наблюдения. В работе [29] рассматривается модель измерения градиента:

$$Y_t = \nabla f_t(\theta) + \xi_t,$$

Для описанного алгоритма, однако измерения градиента не даны, но вдохновляясь работой [10] мы имеем:

$$\bar{Y}_n = \Delta_n \frac{y_n^+ - y_n^-}{2\beta} \tag{25}$$

Разложив функции $f_{2n}(x_n^+)$, $f_{2n-1}(x_n^-)$ по формуле Тейлора, получаем:

$$\begin{gathered} f_{2n}(+) = f_{2n}(x_n^+) = f_{2n}(x_n) + \beta\Delta_n \nabla_{f_{2n}}(\varepsilon_1) \\ f_{2n-1}(-) = f_{2n-1}(x_n^-) = f_{2n-1}(x_n) - \beta\Delta_n \nabla_{f_{2n-1}}(\varepsilon_2) \end{gathered} \tag{26}$$

$$\begin{gathered} \text{где} \quad x_n^+ = x_n + \beta\Delta_n, \quad x_n^- = x_n - \beta\Delta_n \\ \nabla_{f_{2n}}(\varepsilon_1) = \nabla f_{2n}(x_n + \varepsilon_1\beta\Delta) \\ \nabla_{f_{2n-1}}(\varepsilon_2) = \nabla f_{2n-1}(x_n - \varepsilon_2\beta\Delta) \end{gathered} \tag{27}$$

Преобразовав выражение 25, получаем:

$$\begin{gathered} \bar{Y}_n = \Delta_n^T \frac{y_n^+ - y_n^-}{2\beta} = \Delta_n^T \frac{f_{2n}(+) - f_{2n-1}(-) + \xi_{2n} - \xi_{2n-1}}{2\beta} = \\ = \Delta_n^T \frac{f_{2n}(+) - f_{2n}(-) + \varphi_n(-) + \xi_{2n} - \xi_{2n-1}}{2\beta} = \\ = \Delta_n^T \frac{\beta\Delta_n(\nabla_{f_{2n}}(\varepsilon_1) + \nabla_{f_{2n}}(\varepsilon_2))}{2\beta} + \Delta_n^T \frac{\varphi_n(-) + \xi_{2n} - \xi_{2n-1}}{2\beta} \leq \\ \leq \Delta_n^T \frac{\beta\Delta_n(2\nabla f_{2n}(x_n) + L\|\beta\Delta_n\|(\|\varepsilon_1\| + \|\varepsilon_2\|))}{2\beta} + \\ + \Delta_n^T \frac{\varphi_n(-) + \xi_{2n} - \xi_{2n-1}}{2\beta} = \Delta_n^T \frac{\beta\Delta_n 2\nabla f_{2n}(x_n)}{2\beta} + \\ + \Delta_n^T \frac{\beta\Delta_n L\|\beta\Delta_n\|(\|\varepsilon_1\| + \|\varepsilon_2\|)}{2\beta} + \Delta_n^T \frac{\varphi_n(-) + \xi_{2n} - \xi_{2n-1}}{2\beta} = \end{gathered} \tag{28}$$

$$= \nabla f_{2n}(x_n) + \frac{L\|\beta\Delta_n\|(\|\varepsilon_1\| + \|\varepsilon_2\|)}{2} + \Delta_n^T \frac{\varphi_n(-) + \xi_{2n} - \xi_{2n-1}}{2\beta} \tag{29}$$

$$\text{Обозначим,} \quad \zeta_n = \Delta_n^T \frac{\beta\Delta_n(\nabla_{f_{2n}}(\varepsilon_1) + \nabla_{f_{2n}}(\varepsilon_2))}{2\beta} - \nabla f_{2n}(x_n),$$
$$\bar{\xi}_n = \Delta_n^T \frac{\varphi_n(-) + \xi_{2n} - \xi_{2n-1}}{2\beta} \tag{30}$$
$$\bar{Y}_n = \nabla f_{2n}(x_n) + \zeta_n + \bar{\xi}_n$$

Т.к. $\varepsilon_3, \varepsilon_4 \geq 0$ и применив предположение 4: получаем

$$\mathbb{E}\zeta_n = \mathbb{E}\Delta_n^T \frac{\beta\Delta_n(\nabla_{f_{2n}}(\varepsilon_1) + \nabla_{f_{2n}}(\varepsilon_2))}{2\beta} \leq \frac{L\|\beta\|(\|\varepsilon_1\| + \|\varepsilon_2\|)}{2}\mathbb{E}\|\Delta_n\| \leq L\|\beta\|,$$
$$\mathbb{E}\bar{\xi}_n = \mathbb{E}(\Delta_n^T \frac{\varphi_n(-) + \xi_{2n} - \xi_{2n-1}}{2\beta} - \varepsilon_4) = \mathbb{E}\Delta_n^T \frac{\varphi_n(-) + \xi_{2n} - \xi_{2n-1}}{2\beta} = 0 \tag{31}$$

В итоге имеем:

$$\mathbb{E}\bar{Y}_n = \nabla f_{2n}(x_n) + \mathbb{E}\zeta_n \tag{32}$$

**Определение.** Последовательности $\{\lambda_n\}_{n=0}^{\infty}, \lambda_n \geq 0$ и последовательность функций $\{\phi_n(x)\}_{n=0}^{\infty}$ называются $A_n, \Phi$-ограниченными последовательностями оценок для функций $\{f_n(x)\}$, если:

$$\lambda_n \to 0, \tag{33}$$

и существуют такая последовательность $\{A_n\}_{n=0}^{\infty}$, $A_n \in \mathbb{R}$ и константа $\Phi < \infty$ такие, что:

$$\mathbb{E}\phi_n(x) \leq (1 - \lambda_n)f_{2n}(x) + A_n\|\nabla f_{2n}(x)\| + \lambda_n(\tilde{\phi}_{0,n}(x) + \Phi),$$
$$\text{где} \quad \tilde{\phi}_{0,n}(x) = \phi_0(x) - \phi_0(\theta_n) + \phi_0(\theta_0) \tag{34}$$

**Лемма 1.** *Пусть* $\eta \in (0, \mu]$, $\gamma_0 > 0$, $\{x_n\}_{n=0}^{\infty}$ — *произвольная последовательность в* $\mathbb{R}^q$, $\{\alpha_n\}_{n=0}^{\infty}, \{\lambda_n\}_{n=0}^{\infty}$ — *последовательности в* $\mathbb{R}$, *удовлетворяющие (16),* $\{A_n\}_{n=0}^{\infty}$, $\{Z_n\}_{n=0}^{\infty}$ — *последовательности в* $\mathbb{R}$,

*определенные в (17) и* $\{\phi_n(x)\}_{n=0}^{\infty}$ *— последовательность функций, определенная следующим образом:*

$$\phi_0(x) = \phi_0^* + \frac{\gamma_0}{2}\|x - x_0\|^2,$$
$$\phi_{n+1}(x) = (1-\alpha_n)(\phi_n(x) - Z_n) + \alpha_n[r(x, x_n) +$$
$$+\langle \bar{Y}_n(x_n), x - x_n\rangle + (\frac{\mu}{2} - \frac{\eta}{2})\|x - x_n\|^2], \qquad (35)$$
$$\text{где} \quad r(x, x_n) = f_{2n}(x_n) - \frac{c^2}{2\eta} - a\|\nabla f_{2n}(x_n)\| - b - L\beta\|x - x_n\|,$$
$$\mathbb{E}\bar{Y}_n(x_n) = \nabla f_{2n}(x_n) + \mathbb{E}\zeta_n.$$

*Тогда последовательность функций* $\{\phi_n\}_{n=0}^{\infty}$ *является* $A_n, \Phi$ *- ограниченной последовательностью оценок для функций* $\{f_n\}_{n=0}^{\infty}$.

**Доказательство**. По индукции $n \geq 0$. Используя факт того, что $A_0 = 0$, $\lambda_0 = 1$, видим, что для базового случая лемма 1 выполняется:

$$\phi_0(x) = \tilde{\phi}_{0,0}(x) \leq \tilde{\phi}_{0,0}(x) + \Phi.$$

Теперь предполагаем, что для случая $n$ Лемма 1 выполняется, докажем для случая $n+1$, из доказательства в работе [29] имеем:

$$\mathbb{E}\phi_{n+1}(x) = \mathbb{E}[(1-\alpha_n)(\phi_n(x) - Z_n) + \alpha_n[r(x, x_n) + \langle \bar{Y}_n(x_n), x - x_n\rangle +$$
$$+(\frac{\mu}{2} - \frac{\eta}{2})\|x - x_n\|^2]] \leq (1-\alpha_n)(\mathbb{E}\phi_n(x) - Z_n) + \alpha_n f_{2n+2} \leq$$
$$\leq \alpha_n f_{2n+2} + (1-\alpha_n)((1-\lambda_n)f_{2n}(x) + \lambda_n\tilde{\phi}_{0,n}(x) + A_n\|\nabla f_{2n}(x)\| - Z_n).$$

Ввиду того, что $1 - \lambda_{n+1} = 1 - (1-\alpha_n)\lambda_n = (1-\alpha_n)(1-\lambda_n) + \alpha_n$, получаем:

$$\mathbb{E}\phi_{n+1}(x) \leq \alpha_n f_{2n+2} + (1-\alpha_n)((1-\lambda_n)f_{2n}(x) + \lambda_n\tilde{\phi}_{0,n}(x) +$$
$$+A_n\|\nabla f_{2n}(x)\| - Z_n) \leq (1-\lambda_{n+1})f_{2n+2}(x) + \lambda_{n+1}\tilde{\phi}_{0,n}(x) +$$
$$+(1-\alpha_n)((1-\lambda_n)(f_{2n}(x) - f_{2n+2}(x)) + A_n\|\nabla f_{2n}(x)\| - Z_n) \leq$$
$$\leq (1-\lambda_{n+1})f_{2n+2}(x) + \lambda_{n+1}\tilde{\phi}_{0,n}(x) + (1-\alpha_n)\rho(x),$$
$$\rho(x) = (1-\lambda_n)(f_{2n}(x) - f_{2n+2}(x)) + A_n\|\nabla f_{2n}(x)\| - Z_n.$$

Используя Предположение 2, получаем:

$$\tilde{\phi}_{0,n}(x) - \tilde{\phi}_{0,n+1}(x) \leq |\phi_0(\theta_n) - \phi_0(\theta_{n+1})| \leq \frac{\gamma_0}{2}\|\theta_n - \theta_{n+1}\|^2 \leq \frac{\gamma_0 c^2}{2\mu^2}$$

Тогда, применив полученное неравенство в оценке $\mathbb{E}\phi_{n+1}(x)$, получим:

$$\mathbb{E}\phi_{n+1}(x) \leq (1-\lambda_{n+1})f_{2n+2}(x) + \lambda_{n+1}\tilde{\phi}_{0,n}(x) + (1-\alpha_n)\rho(x) \leq$$
$$\leq (1-\lambda_{n+1})f_{2n+2}(x) + \lambda_{n+1}(\tilde{\phi}_{0,n+1}(x) + \Phi) + (1-\alpha_n)\rho(x)$$

Из предположения 2 также получаем:

$$\rho(x) = (1-\lambda_n)(f_{2n}(x) - f_{2n+2}(x)) + A_n\|\nabla f_{2n}(x)\| - Z_n \leq$$
$$\leq (1-\lambda_n)(a\|\nabla f_{2n}(x)\| + b) + A_n\|\nabla f_{2n}(x)\| - Z_n \leq$$
$$\leq (1-\lambda_n)(a\|\nabla f_{2n+2}(x)\| + b) + (1-\lambda_n)ac + A_n\|\nabla f_{2n+2}(x)\| + A_n c - Z_n.$$

Теперь из определения $Z_n$ и $A_n$, получаем:

$$(1-\alpha_n)\rho(x) \leq (1-\alpha_n)[(1-\lambda_n)a + A_n]\|\nabla f_{2n+2}(x)\| +$$
$$+(1-\lambda_n)(b+ac) + A_n c - Z_n =$$
$$= A_{n+1}\|\nabla f_{2n+2}(x)\| + Z_n - Z_n = A_{n+1}\|\nabla f_{2n+2}(x)\|$$

В итоге получаем:

$$\mathbb{E}\phi_{n+1}(x) \leq (1-\lambda_{n+1})f_{2n+2}(x) + \lambda_{n+1}(\tilde{\phi}_{0,n+1}(x) + \Phi) + A_{n+1}\|\nabla f_{2n+2}(x)\|$$

Следовательно последовательность функций $\{\phi_n\}_{n=0}^{\infty}$ является $A_n, \Phi$-ограниченной последовательностью оценок для функций $\{f_n\}_{n=0}^{\infty}$

**Замечание 3** Т.к. $\alpha_j \geq \alpha_x > 0$, $\{A_n\}, \{Z_n\}$ равномерно ограничены в $n$: $A_n < A_\infty < \infty$, $Z_n < Z_\infty < \infty$ и $A_\infty = \frac{a}{\alpha_x}$, $Z_\infty = b + \frac{ac(1+\alpha_x)}{\alpha_x}$.

$$A_{n+1} = (1-\alpha_n)[(1-\lambda_n)a + A_n] \leq (1-\alpha_x)(a + A_n) \leq \frac{a}{\alpha_x}.$$

$$Z_{n+1} = (1-\lambda_{n+1})(b+ac) + A_{n+1}c \leq b + ac\frac{1+\alpha_x}{\alpha_x}.$$

**Лемма 2**. *Функции $\phi_n$, определенные в (35), могут быть выражены*

*в следующей форме:*

$$\phi_n(x) = \phi_n^* + \frac{\gamma_n}{2}\|x - v_n\|^2 \tag{36}$$

$$v_{n+1} = \gamma_{n+1}^{-1}((1-\alpha_n)\gamma_n v_n + \alpha_n(\mu-\eta)x_n - \alpha_n\bar{Y}_n(x_n) + \alpha_n L\beta), \tag{37}$$

$$\gamma_{n+1} = (1-\alpha_n)\gamma_n + \alpha_n(\mu-\eta), \tag{38}$$

$$\begin{gathered}\phi_{n+1}^* = (1-\alpha_n)(\phi_n^* - Z) + \alpha_n r(x_n, x_n) - \\ -\frac{\alpha_n^2}{2\gamma_{n+1}}\|\bar{Y}_n(x_n) - L\beta\|^2 - \alpha_n\frac{(1-\alpha_n)\gamma_n}{\gamma_{n+1}}\langle x_n - v_n, \bar{Y}_n(x_n) - L\beta\rangle + \\ +\frac{1}{2}\frac{(1-\alpha_n)\alpha_n(\mu-\eta)\gamma_n}{\gamma_{n+1}}\|x_n - v_n\|^2.\end{gathered} \tag{39}$$

*Доказательство.* Формулы для $\gamma_n$ вытекают из второй производной выражения (35).

$$\phi_0(x)'' = \gamma_0, \quad \phi_{n+1}(x)'' = \phi_{n+1}^* + \frac{\gamma_{n+1}}{2}\|x - v_{n+1}\|^2 = \gamma_{n+1},$$
$$\phi_{n+1}(x)'' = (1-\alpha_n)\phi_n(x)'' + \alpha_n(\mu-\eta) =$$
$$= (1-\alpha_n)(\phi_n^* + \frac{\gamma_n}{2}\|x - v_n\|^2)'' + \alpha_n(\mu-\eta) =$$
$$= (1-\alpha_n)\gamma_n + \alpha_n(\mu-\eta) = \gamma_{n+1}$$

Если взять градиент $\phi_{n+1}$ и приравнять его к 0, получим:

$$\phi_n(x)' = \gamma_n(x - v_n),$$
$$r(x, x_n)' = -L\beta,$$
$$\phi_{n+1}(x)' = (1-\alpha_n)\phi_n(x)' + \alpha_n r(x, x_n)' + \alpha\bar{Y}_n(x_n) + \alpha_n(\mu-\eta)(x - x_n) =$$
$$= (1-\alpha_n)\gamma_n(x - v_n) - \alpha_n L\beta + \alpha_n\bar{Y}_n(x_n) + \alpha_n(\mu-\eta)(x - x_n),$$
$$(1-\alpha_n)\gamma_n(v_{n+1} - v_n) - \alpha_n L\beta + \alpha_n\bar{Y}_n(x_n) + \alpha_n(\mu-\eta)(v_{n+1} - x_n) = \mathbf{0}$$

из которого вытекает формула для $v_{n+1}$.

$$\begin{gathered}\gamma_{n+1}v_{n+1} = (1-\alpha_n)\gamma_n v_n + \alpha L\beta - \alpha_n \bar{Y}_n(x_n) + \alpha_n(\mu-\eta)x_n, \\ v_{n+1} = \gamma_{n+1}^{-1}((1-\alpha_n)\gamma_n v_n + \alpha_n(\mu-\eta)x_n - \alpha_n\bar{Y}_n(x_n) + \alpha_n L\beta)\end{gathered} \tag{40}$$

Подставив $x_n$ в (35), получим:

$$\phi_{n+1}(x_n) = (1-\alpha_n)(\phi_n(x_n) - Z) + \alpha_n r(x_n, x_n). \tag{41}$$

В то же время, имеем:

$$\phi_{n+1}(x_n) = \phi^*_{n+1} + \frac{\gamma_{n+1}}{2}\|x_n - v_{n+1}\|^2. \tag{42}$$

Используя полученное выражение для $v_{n+1}$ (40), получаем:

$$x_n - v_{n+1} = (\frac{(1-\alpha_n)\gamma_n}{\gamma_{n+1}})(x_n - v_n) + \frac{\alpha_n}{\gamma_{n+1}}(\bar{Y}_n(x_n) - L\beta), \tag{43}$$

так, что

$$\begin{gathered}\frac{\gamma_{n+1}}{2}\|x_n - v_{n+1}\|^2 = \frac{\gamma_{n+1}}{2}(\frac{(1-\alpha_n)\gamma_n}{\gamma_{n+1}})^2\|x_n - v_n\|^2 + \\ +2\frac{\gamma_{n+1}}{2}(\frac{(1-\alpha_n)\gamma_n}{\gamma_{n+1}})\frac{\alpha_n}{\gamma_{n+1}}\langle x_n - v_n, \bar{Y}_n(x_n) - L\beta\rangle + \\ +\frac{\gamma_{n+1}}{2}(\frac{\alpha_n}{\gamma_{n+1}})^2\|\bar{Y}_n(x_n) - L\beta\|^2 = \frac{(1-\alpha_n)^2\gamma_n^2}{2\gamma_{n+1}}\|x_n - v_n\|^2 + \\ +(1-\alpha_n)\gamma_n\frac{\alpha_n}{\gamma_{n+1}}\langle x_n - v_n, \bar{Y}_n(x_n) - L\beta\rangle + \frac{\alpha_n^2}{2\gamma_{n+1}}\|\bar{Y}_n(x_n) - L\beta\|^2.\end{gathered} \tag{44}$$

Вычитая (42) из (41) получаем формулу для вычисления $\phi^*_{n+1}$

$$\begin{gathered}(1-\alpha_n)(\phi_n(x_n) - Z) + \alpha_n r(x_n, x_n) - \phi^*_{n+1} - \frac{\gamma_{n+1}}{2}\|x_n - v_{n+1}\|^2 = \\ = (1-\alpha_n)(\phi^*_n + \frac{\gamma_n}{2}\|x_n - v_n\|^2 - Z) + \alpha_n r(x_n, x_n) - \\ -\phi^*_{n+1} - \frac{\gamma_{n+1}}{2}\|x_n - v_{n+1}\|^2 = 0\end{gathered}$$

$$\phi_{n+1}^* = (1-\alpha_n)(\phi_n^* - Z) + \alpha_n r(x_n, x_n) +$$
$$+(1-\alpha_n)\frac{\gamma_n}{2}\|x_n - v_n\|^2 - \frac{\gamma_{n+1}}{2}\|x_n - v_{n+1}\|^2 =$$
$$= (1-\alpha_n)(\phi_n^* - Z) + \alpha_n r(x_n, x_n) +$$
$$+((1-\alpha_n)\frac{\gamma_n}{2} - \frac{(1-\alpha_n)^2\gamma_n^2}{2\gamma_{n+1}})\|x_n - v_n\|^2 -$$
$$-(1-\alpha_n)\gamma_n\frac{\alpha_n}{\gamma_{n+1}}\langle x_n - v_n, \bar{Y}_n(x_n) - L\beta\rangle - \frac{\alpha_n^2}{2\gamma_{n+1}}\|\bar{Y}_n(x_n) - L\beta\|^2 =$$
$$= (1-\alpha_n)(\phi_n^* - Z) + \alpha_n r(x_n, x_n) - \frac{\alpha_n^2}{2\gamma_{n+1}}\|\bar{Y}_n(x_n) - L\beta\|^2 -$$
$$-\alpha_n\frac{(1-\alpha_n)\gamma_n}{\gamma_{n+1}}\langle x_n - v_n, \bar{Y}_n(x_n) - L\beta\rangle +$$
$$+(1-\alpha_n)\frac{\gamma_n}{2}(1 - \frac{(1-\alpha_n)\gamma_n}{\gamma_{n+1}})\|x_n - v_n\|^2 =$$
$$= (1-\alpha_n)(\phi_n^* - Z) + \alpha_n r(x_n, x_n) - \frac{\alpha_n^2}{2\gamma_{n+1}}\|\bar{Y}_n(x_n) - L\beta\|^2 -$$
$$-\alpha_n\frac{(1-\alpha_n)\gamma_n}{\gamma_{n+1}}\langle x_n - v_n, \bar{Y}_n(x_n) - L\beta\rangle +$$
$$+\frac{1}{2}\frac{(1-\alpha_n)\alpha_n(\mu-\eta)\gamma_n}{\gamma_{n+1}}\|x_n - v_n\|^2$$

Лемма 3 формулируется и доказывается анлалогично с [29]:

**Лемма 3.** *(See [19], 2.2.1). Пусть* $\{\lambda_n\}, \{\phi_n(x)\} - A_n, \Phi$*-ограниченные последовательности оценок для функций* $\{f_n(x)\}$ *и для некоторой последовательности* $\{theta_n\}_{n=0}^{\infty}$ *в* $\mathbb{R}^q$, $\{D_n\}_{n=0}^{\infty}$ *in* $\mathbb{R}$, $D_n \geq 0$, $D_n < D_\infty < \infty$ *для всех* $n \geq 0$ *выполняются следующие неравенства:*

$$\mathbb{E}_n f_n(\theta_n) \leq \phi_n^* + D_n = \min_{x\in\mathbb{R}^q}\phi_n(x) + D_n, \qquad (45)$$

*тогда*

$$\mathbb{E}_n f_n(\theta_n) - f_n^* \leq \lambda_n(\phi_0(\theta_0) - f^* + \Phi) + D_n \to_{n\to\infty} D_\infty. \qquad (46)$$

*Доказательство.*

$$\mathbb{E}_n f_n(\theta_n) \leq \phi_n^* + D_n \leq$$
$$\leq \phi_n(\theta_n) + D_n \leq (1-\lambda_n)f_n(\theta_n) + \lambda_n\tilde{\phi}_{0,n}(\theta_n) + \lambda_n\Phi + D_n$$

Используя $\tilde{\phi}_{0,n}(x) = \phi_0(x) - \phi_0(\theta_n) + \phi_0(\theta_0)$, получаем:

$$\mathbb{E}_n f_n(\theta_n) \leq (1 - \lambda_n) f_n(\theta_n) + \lambda_n \phi_0(\theta_0) + \lambda_n \Phi + D_n,$$

$$\mathbb{E}_n f_n(\theta_n) - f_n^* \leq \lambda_n (\phi_0(\theta_0) - f_n(\theta_n)) + \lambda_n \Phi + D_n.$$

# 3. Системы

## 3.1. Обзор существующих систем

На сегодняшний день существует немалое количество инструментов, предназначенных для решения задач математической оптимизации [1, 32, 21, 17, 26, 8, 16, 30, 9]. Список инструментов математической оптимизации включает в себя как бесплатные, так и проприетарное программное обеспечение (ПО), как программные приложения, так и программные библиотеки, причем значительную часть составляют библиотеки и проприетарное ПО, однако все они не подходят для достижения описываемой в работе цели ввиду нескольких причин.

Программные библиотеки, к примеру, не обладают графическим интерфейсом и в большенстве своем привязаны к конкретному языку программирования, что существенно усложняет разработку новых алгоритмов людям, не знакомым с такими яыками программирования. Например библиотеки SciPy и Gekko используются только в Python. Недостаток проприетарного ПО очевиден — такие инструменты, как ALGLIB[1], GAMS[2], MIDACO[3], TOMLAB[4], MOSEK[5], Optimization Tool Matlab[6], IMSL Numerical Libraries[7] и др. не доступны в свободном использовании и для них нужно покупать дополнительную лицензию. Также большинство этих систем не предоставляют функции сравнения и разработки новых алгоритмов, они дают только возможность использования уже реализованных алгоритмов, как например IPOPT[8], ALGLIB, GAMS, SciPy[9], Gekko[10] и др. Однако главным недостатком всех существующих систем оптимизации является то, что они не предназначены для задач трекинга и работают только для случая, когда

[1] https://www.alglib.net/docs.php
[2] https://www.gams.com
[3] http://www.midaco-solver.com/index.php/about
[4] https://tomopt.com/tomlab/about/
[5] https://www.mosek.com/documentation/
[6] https://www.mathworks.com/help/optim/
[7] https://www.imsl.com
[8] https://github.com/coin-or/Ipopt
[9] https://www.scipy.org/about.html
[10] https://gekko.readthedocs.io/en/latest

данные стационарные.

| Система | GUI | Пользов. алгоритм | Модель измерений | Модель шума | Модель дрифта | Итерац. и шаги |
|---|---|---|---|---|---|---|
| Optimization Tool Matlab | + | — | + | — | — | + |
| ADMB | — | + | + | + | — | + |
| GAMS | — | + | + | — | — | + |
| CUTEr | — | + | + | — | — | + |
| Новая система | + | + | + | + | + | + |

Таблица 1: Сравнение систем

В Таблице 1 показаны преимущества и недостатки систем моделирования, которые больше всего подходят для достижения цели данной работы. Все системы, как было сказано ранее, рашают задачу стационнарной оптимизации и не подходят для задач трекинга. Самой близкой системой, к разрабатываемой является Optimization Tool Matlab, в ней, также имеется графический интерфейс, в то время как в остальных системах CUTEr[11], GAMS и ADMB[12] графический интерфейс отсутствует. Однако, также как и все другие методы, она решает задачи стационнарной оптимизации и не предназначена для решения задач трекинга. К тому же в Optimization Tool не предусмотрена возможность задания пользовательских алгоритмов, вследствии чего она может применяться для сравения существующих методов, но не может быть использована в разработке новых. Еще один недостаток состоит в том, что в большинстве систем нет возможности задания модели шума и даже в тех системах, в которых она есть, к примеру в ADMB, данная возможность весьма ограничена, т.е. в ней есть только возможность задания случайных помех.

[11] `https://en.wikipedia.org/wiki/CUTEr`

[12] `http://www.admb-project.org`

## 3.2. Система для моделирования алгоритмов стохастической оптимизации для задач трекинга

### 3.2.1. Описание системы

Как было ранее сказано, система предназначена для моделирования алгоритмов стохастической оптимизации для задач трекинга. Главным преимуществом описываемой системы, выделяющим ее среди других, ранее описанных систем, являются ее требования. Их реализация предоставляет пользователям возможность сравнивать, визуализировать и тестировать алгоритмы оптимизации, тем самым облегчая разработку новых алгоритмов и предоставляя пользователям устройство для выбора наиболее эффективного алгоритма для конкретной задачи. В то время как удобный графический интерфейс системы создает необходимые условия для взаимодействия с самой системой.

**Требования к системе.** В процессе создания системы были выдвинуты следующие требования:

- Возможность задания модели измерений.
  Поскольку в системе есть алгоритмы, использующие измерения функций, и алгоритмы, использующие измерения градиента, должна быть предусмотрена возможность задания как градиента функции, так и самой функции.
- Возможность задания модели шума.
- Возможность задания модели дрифта или модели изменения точки минимума.
- Возможность добавления пользовательских алгоритмов.
- Возможность задания количества шагов и количества итераций алгоритма. Т.к. система предназначена для задач трекинга, то

должна быть предусмотрена возможность задания количества шагов алгоритма, которая определяет сколько моментов времени $n$ (2) рассматривается в системе, а также возможность задания количества итераций, по которым вычисляется усредненное значение среднеквадратичных ошибок.

**Графический интерфейс системы.** Внешний вид системы показан на Рис.1 Разберём подробнее реализацию вышеописанных требований на графическом интерфейсе системы.

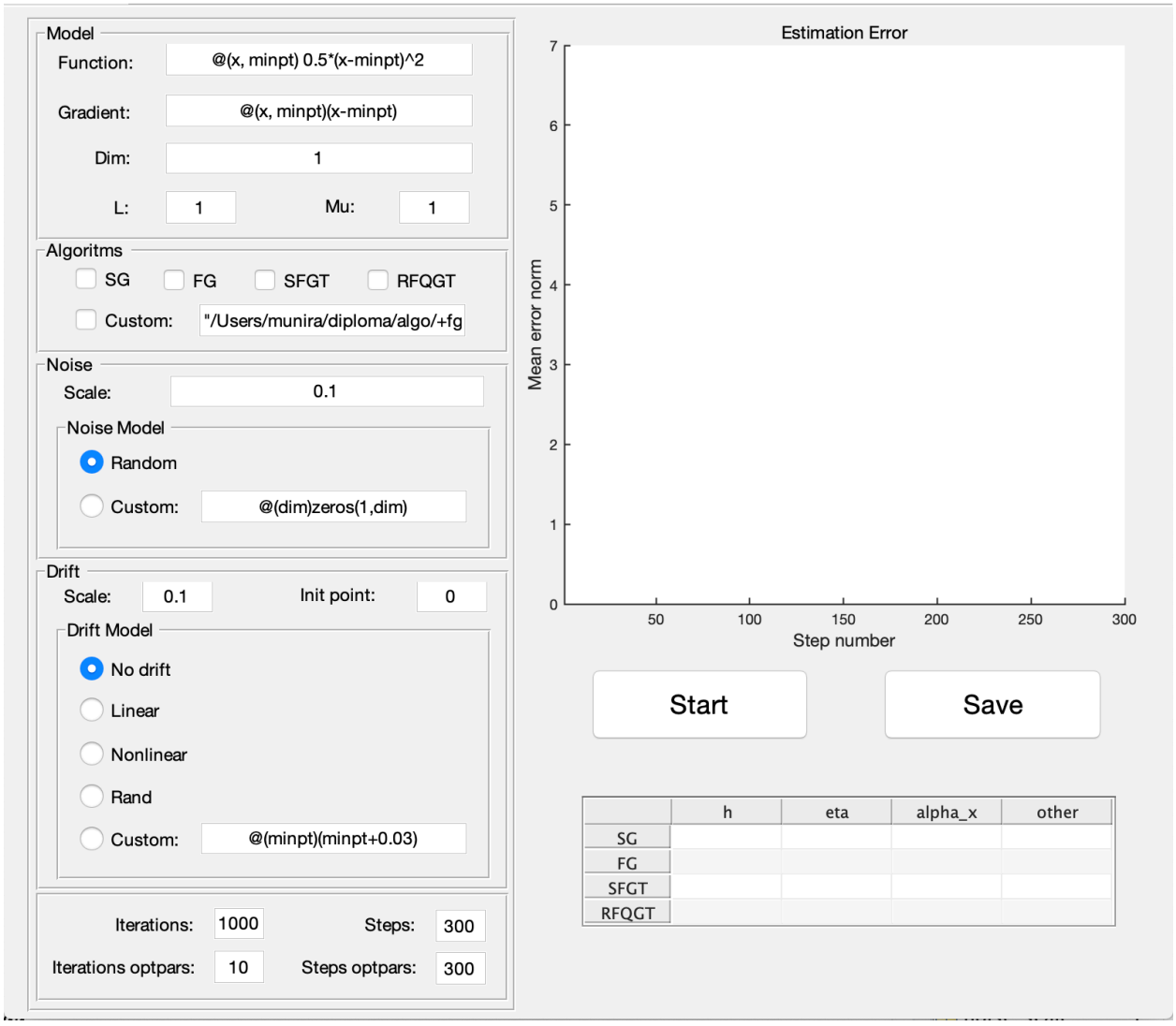


Рис. 1: Графический интерфейс системы

Для кадого из вышеописанных требований в графическом интерфейсе системы выделен отдельный блок. Так, к примеру, для задания

модели измерений нужно ввести описание функции, градиента, размерность модели, константу Липшица и константу строгой выпуклости в соответствующие поля в блоке “Model”. Все функции, а именно градиент функции измерений, сама функция измерений, функция дрифта и шума, задаются в виде анонимных Matlab функций, как показано на Рис.1.

Для того, чтобы определить какие алгоритмы будут использовться в системе необходимо выбрать соответствующие алгоритмы в блоке “Algorithms”, в нем флажки SG[13], FG[14], SFGT[15] и RFQGT[16] — обозначают алгоритмы «Стохастический градиентный спуск», «Быстрый градиентный метод Нестерова», «Быстрый стохастический градиентный метод для задач трекинга» и «Рандомизированный квази-градиентный метод для задач трекинга» соответственно. Для задания пользовательского алгоритма, нужно ввести путь к директории с функциями алгоритма в поле “Custom”.

Модель шума можно определить в блоке “Noise”, для этого нужно определить шкалу шума в поле “Scale” и выбрать модель шума — случайный “Random” или пользовательский “Custom”. На Рис.1 в поле “Custom” блока “Noise” показано как можно задать нулевой шум с помощью пользовательской модели шума.

Модель дрифта или модель изменения точки минимума, задаётся в блоке “Drift”. В системе существует возможнось выбора одной из моделей дрифта: отсутствие дрифта “No drift”, линейный дрифт “Linear”, нелинейный дрифт “Nonlinear”, случайный “Rand” и пользовательский “Custom”. Также есть возможность задать шкалу шума и начальную точку минимума $\theta_0$ в (2). Формулы вычисления точки минимума для всех случаев дрифта представлены ниже:

$$\theta_0 = 0$$
$$\theta_n = \theta_{n-1} + t_n.$$

[13]Стохастический градиентный спуск (Глава 2.2)

[14]Быстрый градиентный метод Нестерова (Глава 2.3)

[15]Быстрый стохастический градиентный метод для задач трекинга (Глава 2.4)

[16]Рандомизированный квази-градиентный метод для задач трекинга (Глава 2.5)

- Остутствие дрифта

$$\bar{t}_n = 0$$

- Линейный дрифт

$$\bar{t}_n = d\frac{\bar{t}}{\|\bar{t}\|}$$

- Случайный дрифт

$$\bar{t}_n \in N(0, I), \quad t_n = d\frac{\bar{t}_n}{\|\bar{t}_n\|}$$

- Нелинейный дрифт

$$m(n) = \mod(n, 100),$$
$$\zeta_n = 0.01m(n)\zeta_1 + (1 - 0.01m(n))\zeta_2$$
$$\bar{t}_n = d\frac{\zeta_n}{\|\zeta_n\|}$$

Задать количество итераций и количество шагов алгоритмов можно в полях “Iterations” и “Steps” соответственно. Поля “Iterations optpars” и “Steps optpars” определяют количество итераций и шагов для поиска оптимальных параметров.

Кнопка “Start” запускает моделирование, по окончанию которого результаты работы алгоритмов в виде сравнительного графика и таблицы оптимальных параметров будут выведены в соответствующие места графического интерфейса. Также кнопка “Start” выводит результирующий график в отдельном окне, которое дает пользователям возможность более детально изучить график и сохранить его в нужное место. Кнопка “Save” предоставляет возможность сохранить найденные оптимальные параметры и параметры запуска модели — модель измерений, модель дрифта и шума и т.д. в формате MatFile.

### 3.2.2. Архитектура системы

Для создания системы использовались следующие программные средства:

- Matlab[17] — система написана на Matlab.
- Matlab App Designer[18] — для создания графического интерфейса системы.
- Matlab Application Compiler[19] — для создания независимого приложения.

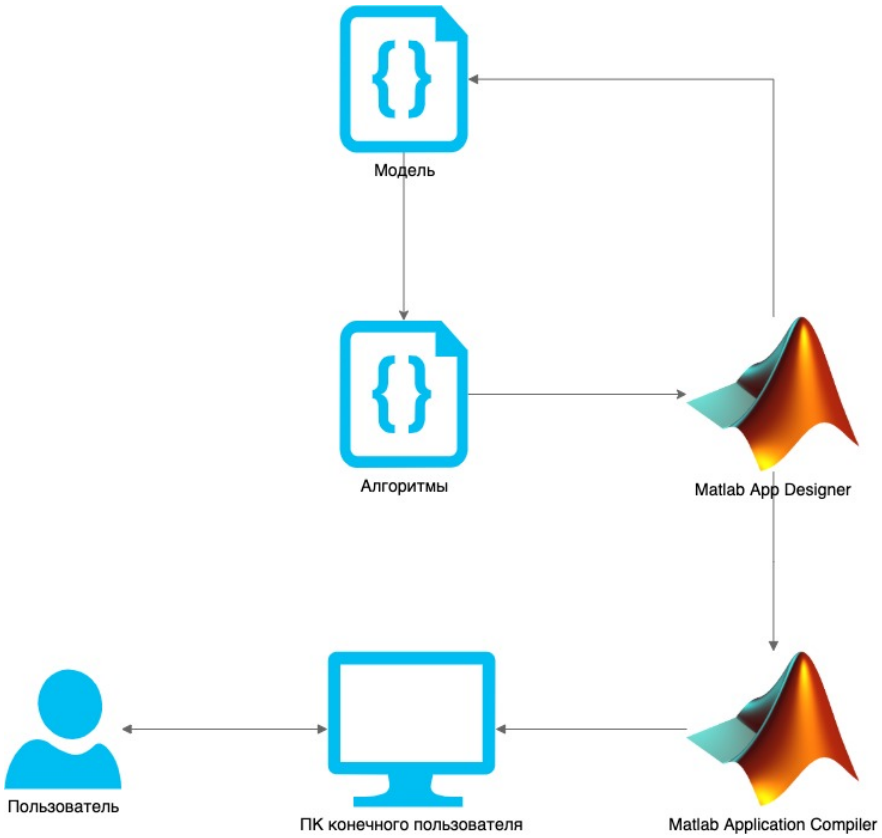


Рис. 2: Архитектура системы

Система представляет собой независимое пользовательское приложение, архитектура которого показана на Рис.2. Модель получает данные, введенные пользователем в графическом интерфейсе, от Matlab

[17] https://www.mathworks.com
[18] https://www.mathworks.com/products/matlab/app-designer.html
[19] https://www.mathworks.com/products/compiler.html

App Designer. Затем, уже сформированная модель подается на вход алгоритмов, где вычисляются оптимальные параметры алгоритмов, оценки точек минимума и значения среднеквадратичных ошибок оценок. Далее, результаты работы алгоритмов, а именно результирующий график среднеквадратичных ошибок оценок и значения оптимальных параметров, принимает Matlab App Designer и выводит результаты работы алгоритмов в аграфическом интерфесе системы. С помощью Matlab Application Compiler упаковываются все необходимые библиотеки и в итоге имеем независимое приложение, которое не требует установки Matlab для запуска на ПК конечного пользователя.

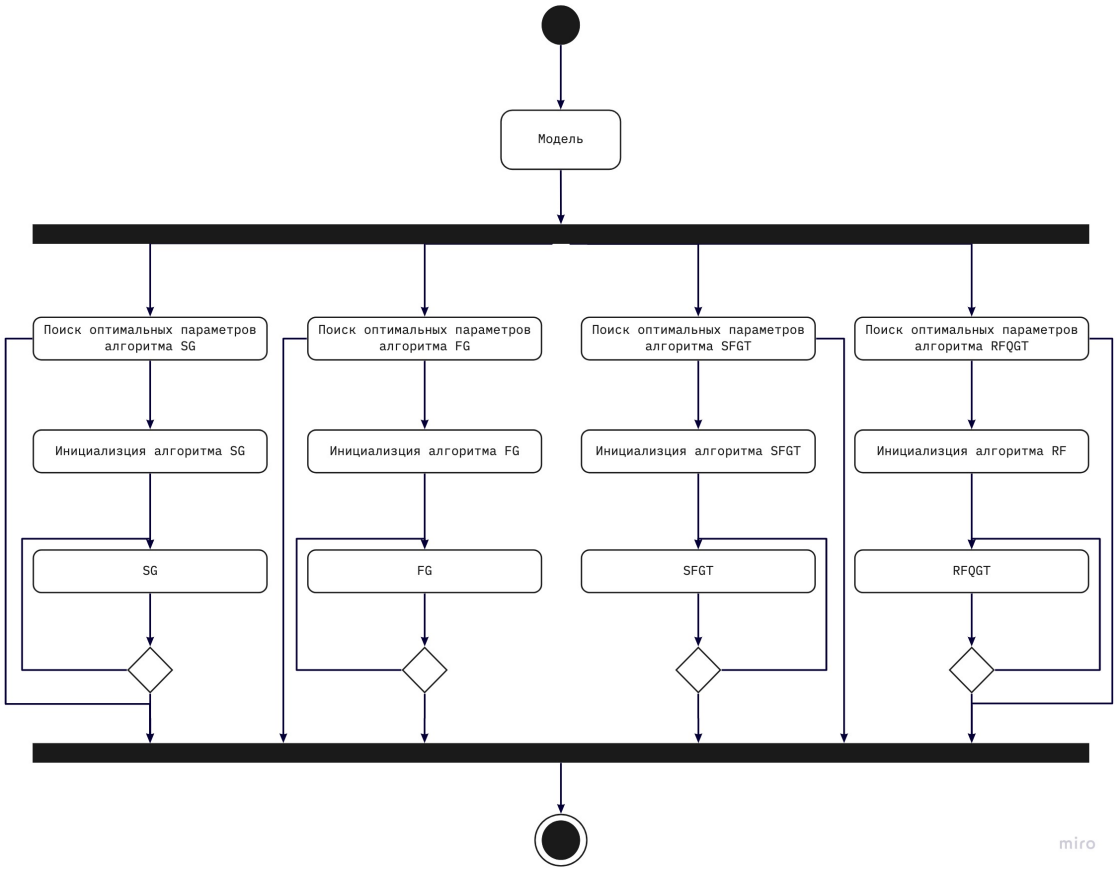


Рис. 3: Диаграмма активностей системы

Общая структура работы в системе изображена на Рис.3 в виде диаграммы активностей. Системе подаются на вход параметры, заданные в графическом интерфейсе, из которых впоследствии строится модель. Дальше для каждого алгоритма вычисляются оптимальные параметры, путем прогона каждого метода на различных параметрах алгорит-

ма. После, для каждого метода происходит инициализация самого алгоритма. И далее итеративно вычисляется значение оценки точки минимума. Делается прогон в заданное количество итераций и для каждого метода усредняется значение среднеквадратичных ошибок. И в итоге получаем результат в виде графика среднеквадратичных ошибок оценок алгоритмов и оптимальных параметров.

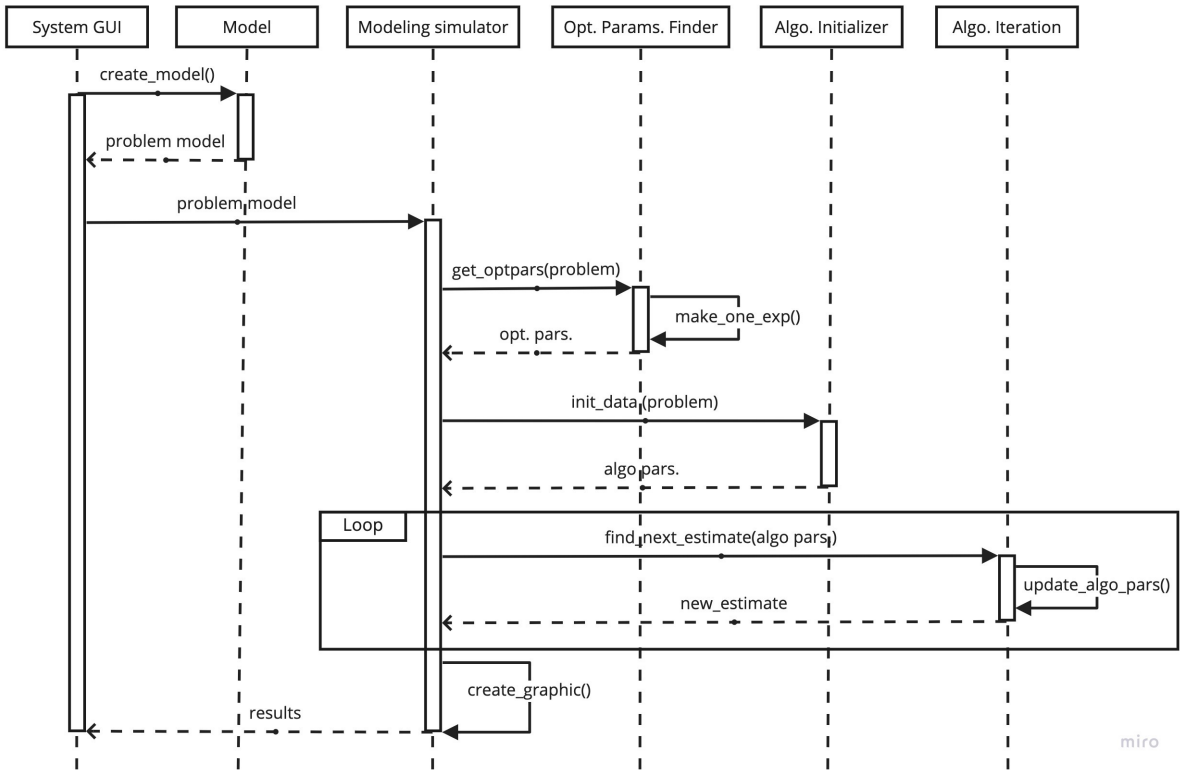


Рис. 4: Диаграмма последовательности системы

На Рис.4 изображена диаграмма последовательности системы. Модель задачи, полученная из введеных пользователем данных в графичесвом интерфейсе системе, подается на вход в симулятор модели, который запускает моделирование. Далее симулятор находит оптимальные параметры для каждого алгоритма, реализованного в модели, с помощью функции `get_optpars()`. В этой функции берется множество теоретически оптимальных параметров и для каждого набора параметров вычисляется среднее по числу итераций значение среднеквадратичной ошибки алгоритма при помощи функции `make_one_exp()`. В результате выбирается тот набор параметров, который дает наилучшую оценку. Далее вызывается функция `init_data()`, которая вычисляет

остальные параметры алгоритмов (второй шаг в описании алгоритмов в Главе 2) и возвращает структуру алгоритма с параметрами. Затем, в цикле вычисляется последовательность оценок определенного алгоритма с помощью функции `find_next_estimate`, которая обновляет параметры алгоритма на каждой шаге и возвращает значение следующей оценки $\theta_{n+1}$. Получив значения оценок алгоритмов, симулятор вычисляет значения соответствующих среднеквадратичных ошибок и возвращает полученные результаты для вывода в графическом интервейсе системы. Важно отметить, что пользовательские алгоритмы также должны быть реализованы подобным образом, т.е. в директории с пользовательским алгоритмом, указанной в графической интерфейсе системы, должны быть реализованы следующие матлаб функции: `get_optpars()`, `init_data()` и `find_next_estimate`.

### 3.2.3. Апробация в системе

В ходе апробации системы были протестированы все возможности системы — были протестированы различные модели измерений, различные модели шума и дрифта, различное количество итераций и шагов алгоритма, возможность сохранения результатов запуска моделирования и параметров, а также были протестированы различные алгоритмы и возможность добавления новых. Также, в результате апробации системы были получены сравнительные результаты времени работы алгоритмов для различных моделей измерений, шума и дрифта. Далее будут показаны результаты работы системы на некоторых примерах.

Первый пример — это случай отсутствия шума и дрифта, т.е. случай стационарной детерминированной оптимизации. (Рис.5) Нулевой шум задан с помощью пользовательской анонимной матлаб функции. В качестве модели функций взята квадратичная функция единичной размерности. В этом примере молелировния рассматриваются 300 измерений (т.к. число шагов в алгоритме равно 300) и значение ошибки для каждого шага усредняется по 100 измерениям. Использованные методы для данного случая — это метод стохастического градиента(SG), быстрый градиентный метод Нестерова(FG), быстрый стохастический

градиентный метод (SFGT) и новый метод (RFQGT).

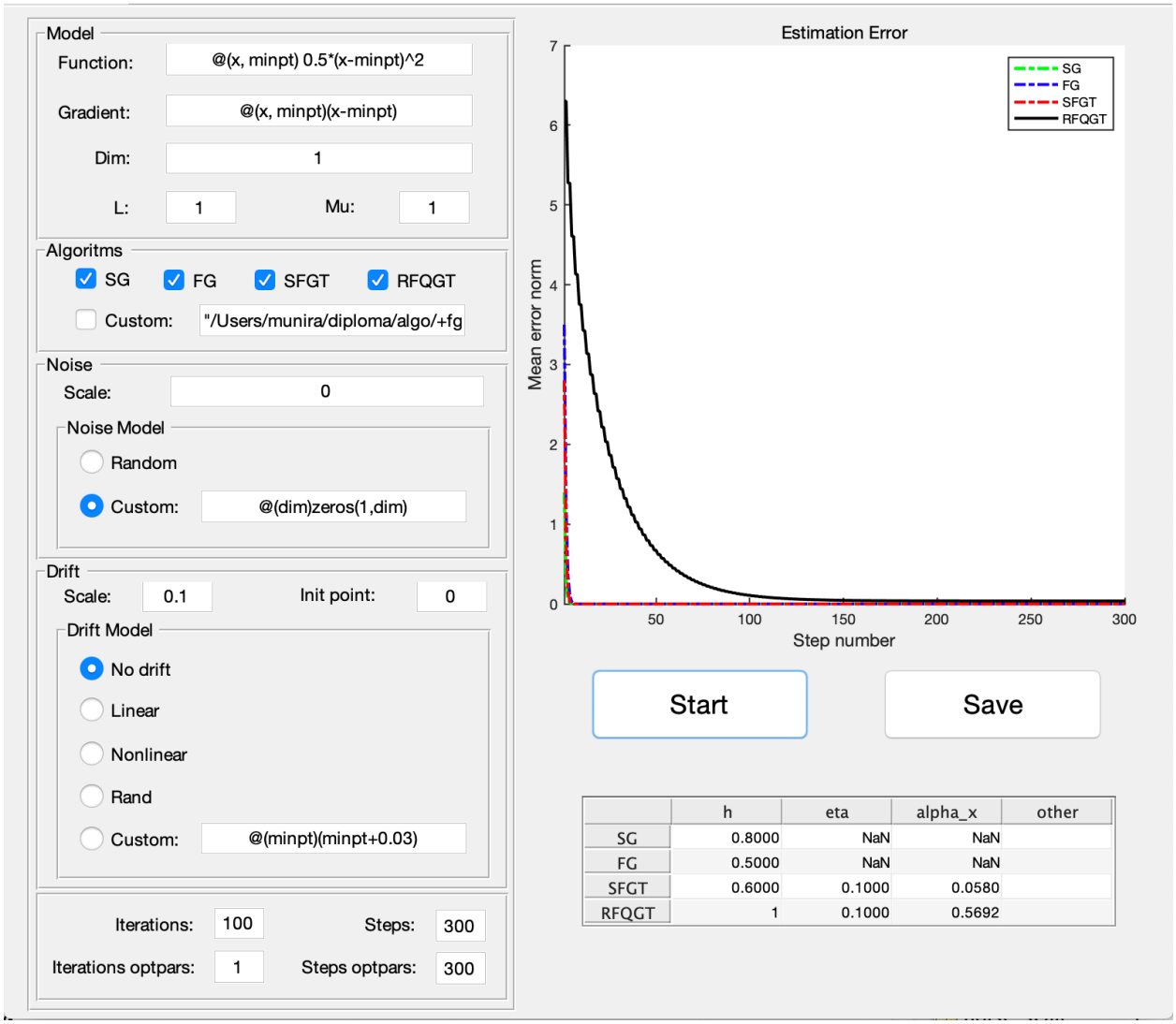


Рис. 5: Случай отсутствия шума и дрифта

Следующий пример — это случай с нелинейным дрифтом со шкалой 0.1 и со случайным шумом со шкалой 0.1. (Рис.6) В этом примере также взята квадратичная функция единичной размерности в качестве модели функций. Алгоритмы, количество итераций и количество шагов такие же как и в предыдущем примере.

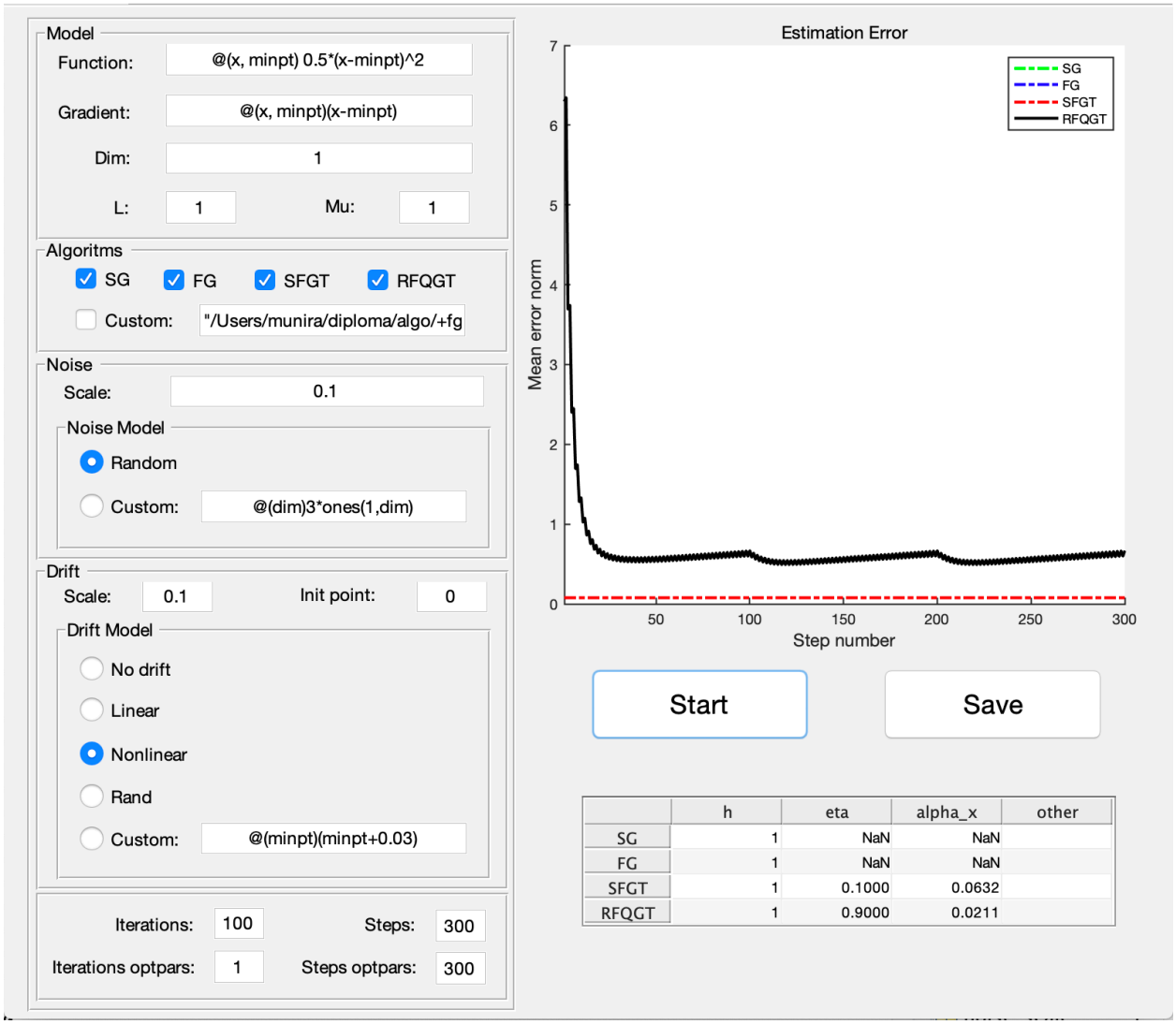


Рис. 6: Нелинейный дрифт, случайный шум

В следующем примере рассматривается случай, схожий с предыдущим за исключением того, что тут рассматривается большая шкала шума — в предыдущем примере шкала шума была равна 0.1, в этом — она равна 1. (Рис.7) Этот и предыдущий пример показывают, что новый алгоритм слабо чувствителен к шкале шума, следовательно, при увеличении значения шума, оценка параметра $\theta_n$, полученная новым алгоритмом, в отличии от других методов, не сильно ухудшается. В связи с этим, на больших значениях шума новый алгоритм дает лучшие оценки, чем другие градиентные методы, реализованные в системе.

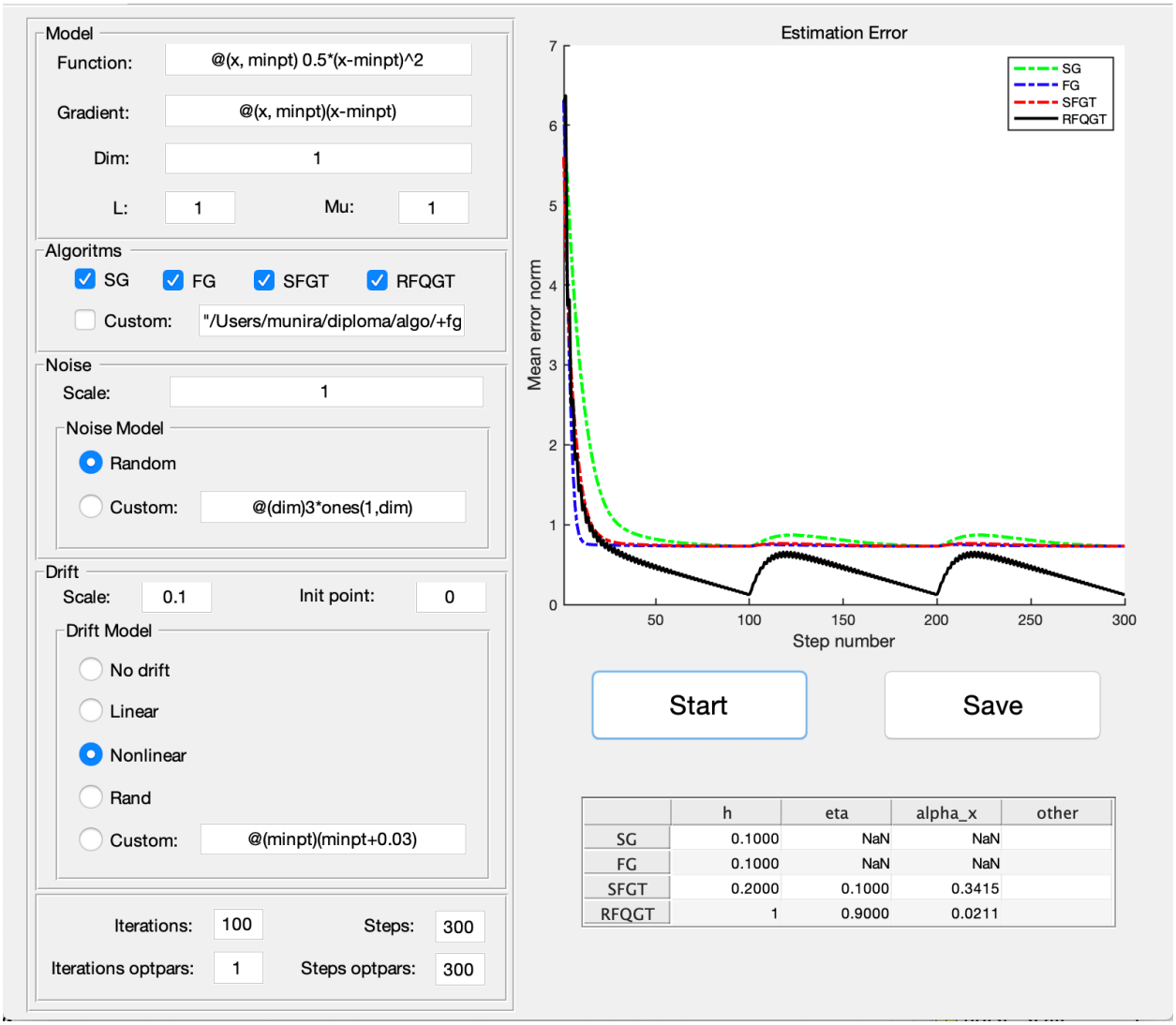


Рис. 7: Нелинейный дрифт, случайный шум

На Рис.8 изображен пример, демонстрирующий возможность добавления пользовательских алгоритмов в систему. В нём рассматривается случай случайного дрифта со шкалой 0.1 и случайного шума со шкалой 0.1. В этом примере взята квадратичная функция единичной размерности, в нем рассматриваются 300 измерений и 100 итераций. Использованные методы — это метод стохастического градиента(SG), быстрый стохастический градиентный метод (SFGT), новый метод (RFQGT) и пользовательский алгоритм. В качестве пользовательского алгоритма был реализован быстрий градиентный метод Нестерова.

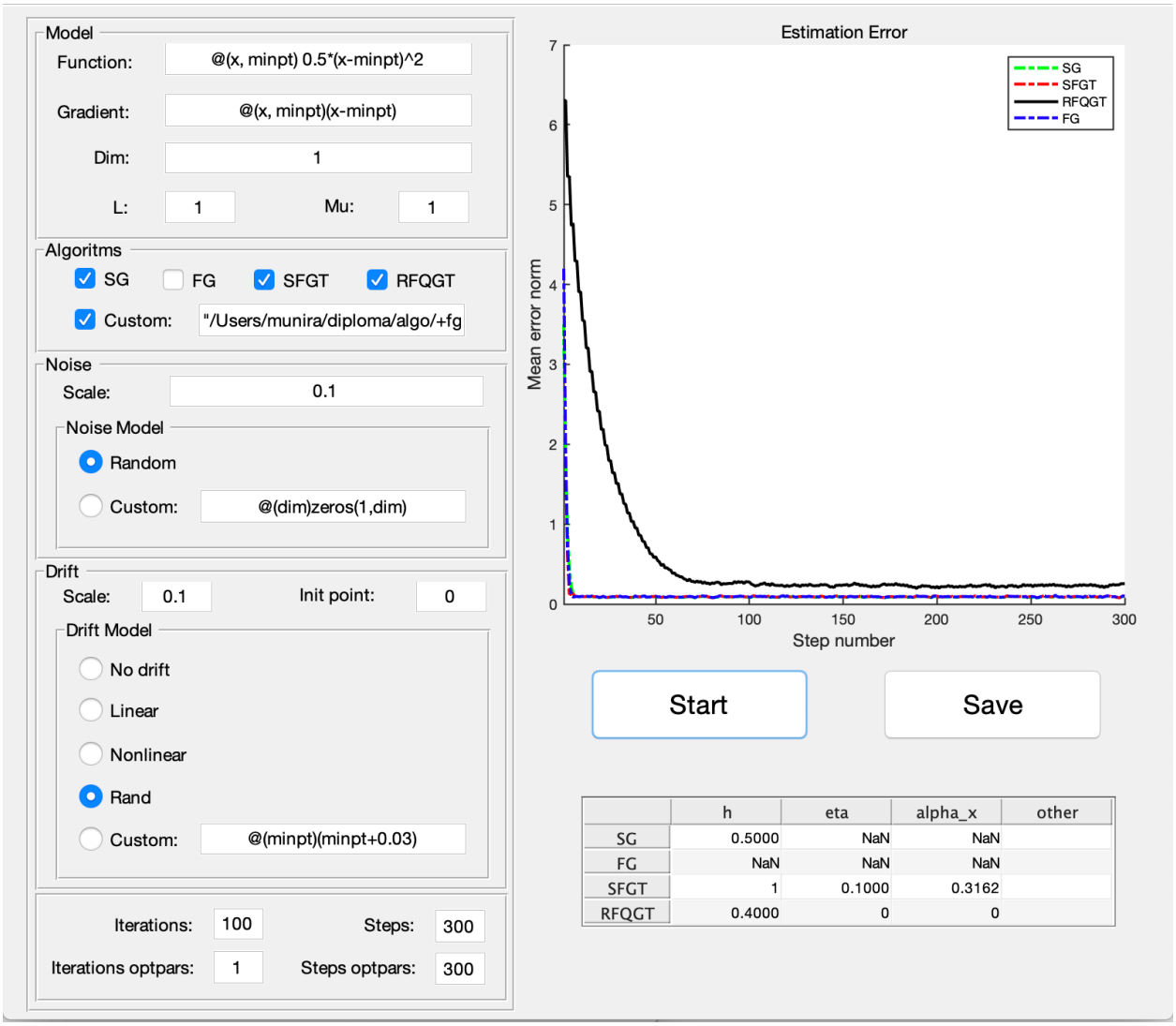


Рис. 8: Пользовательский алгоритм

Другой пример запуска моделирования, изображенный на Рис.9, показывает возможность добавления пользовательской модели шума. В этом примере, также как и в первом, рассматриваются 300 измерений, 100 итераций и алгоритмы SG, FG, SFGT, RFQGT. Модель дрифта линейная, со шкалой 0.1, а пользовательская модель шума задает константный шум со шкалой 1. Как видно из графика (Рис.9), новый метод (RFQGT) дает наилучшую оценку, поскольку значение шума достаточно большое и, как было сказано ранее, новый метод (RFQGT) слабо чувствителен к шкале шума.

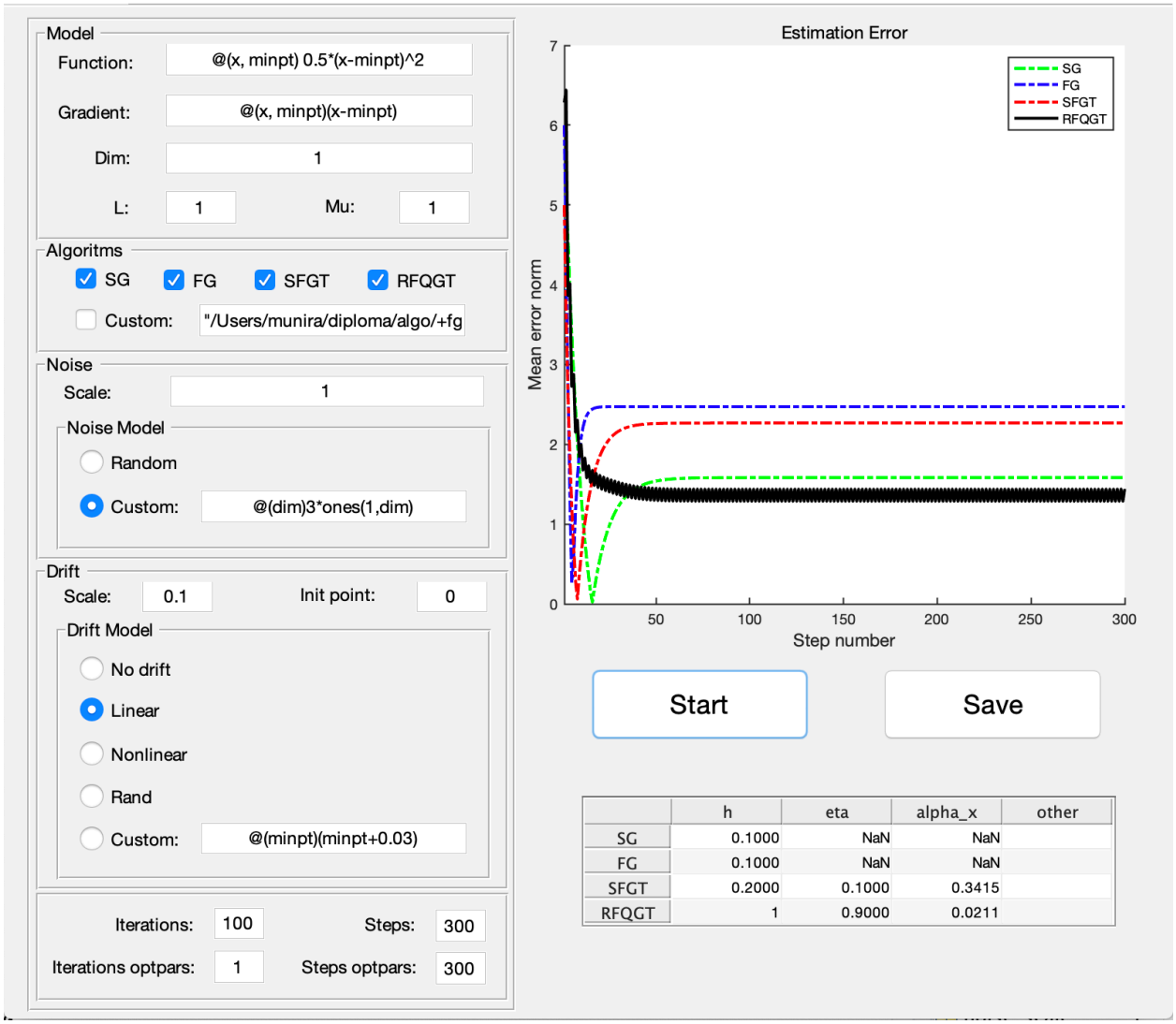


Рис. 9: Пользомательская модель шума

В примере запуска моделирования, изображенном на Рис. 10, показана возможность добавления пользовательской модели дрифта. Тут также рассматриваются 300 измерений, 100 итераций и все алгоритмы, реализованные в системе. Модель шума — случайная со шкалой 0.1. Точка минимума в этом примере изменяется по следующему закону:

$$\theta_{n+1} = \theta_n + 0.03 * 0.1$$

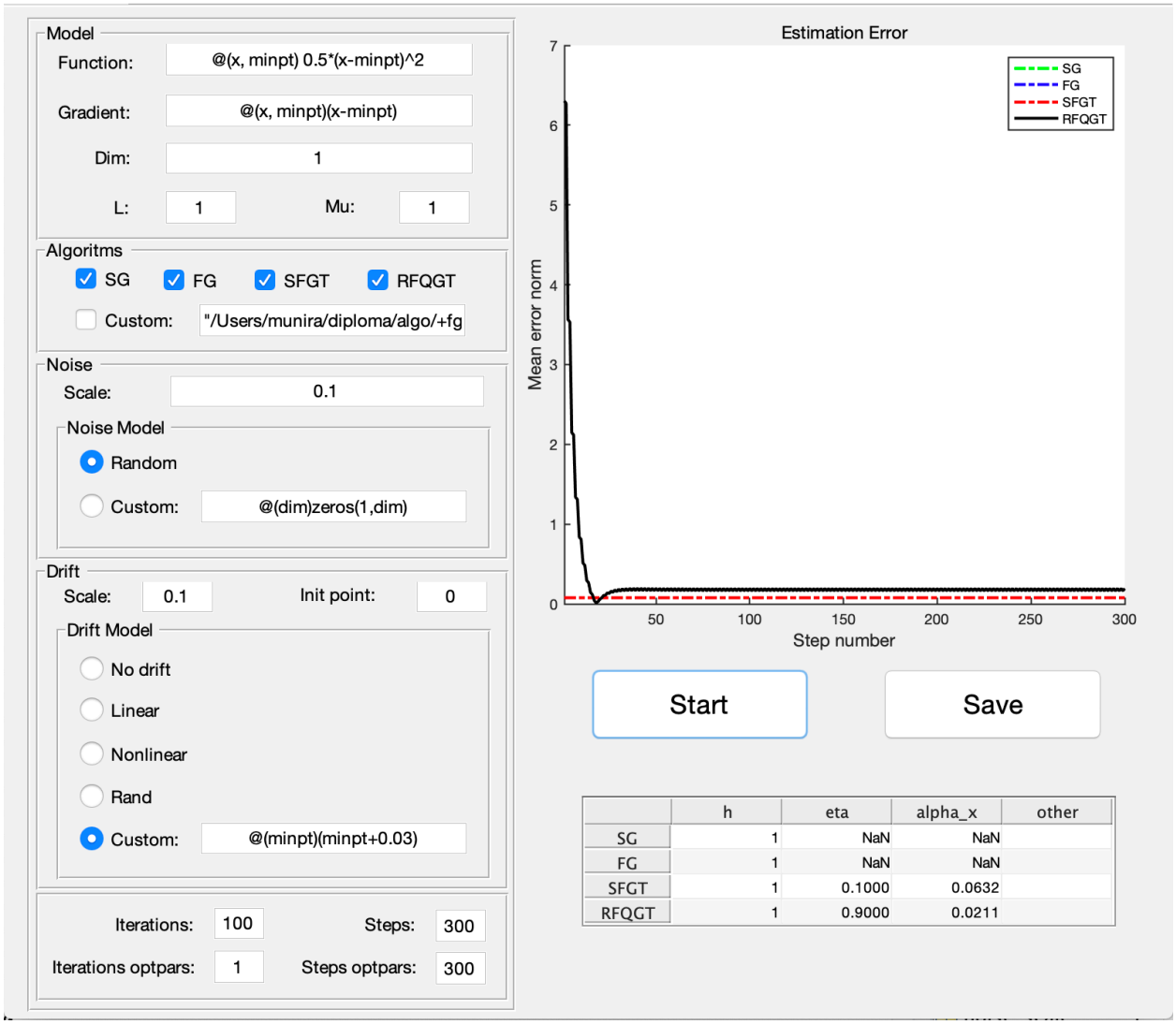


Рис. 10: Пользовательская модель дрифта

В предыдущих примерах рассматривалась квадратичная функция единичной размерности, в этом примере рассматривается квадратичная функция размерности 3. Количество итераций тут также равно 100, а количество шагов 300. Также, во всех предыдущих примерах рассматривалась шкала дрифта, равная 0.1, в примере, проиллюстрированном на Рис.11, рассматривается случайный дрифт, со шкалой 0.01 и случайным шум, со шкалой 1. Из результатов моделирования видно, что в этом примере новый алгоритм (RFQGT) также дает наилучшие результаты, в связи с тем, что значение шума достаточно велико.

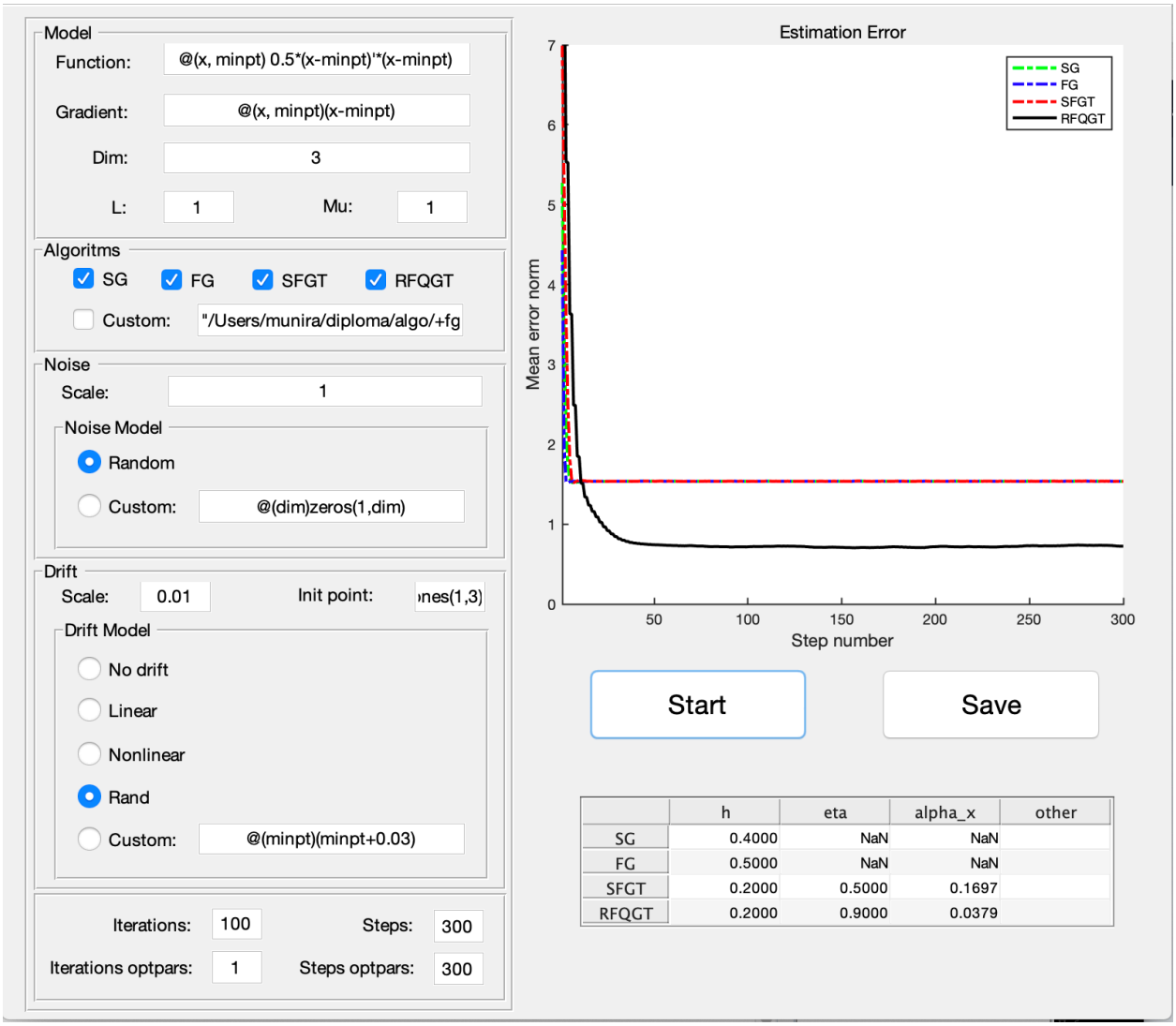


Рис. 11: Модель измерений размерности 3

Последний случай проиллюстрирован на Рис.12. Он схож с предыдущим за исключением того, что тут рассматривается большая шкала дрифта и больше итераций для сглаживания результирующего графика, а именно шкала дрифта равна 0.1, а количество итераций равно 1000.

Все вышеописанные примеры демонстрируют различные возможности системы. Из полученных результатов можно сделать следующие выводы: Во-первых, новый алгоритм (RMFQGT) разумно применять в моделях маленьких размерностей, с небольшим изменением дрифта, поскольку он чувствителен к величине дрифта и размерности модели.

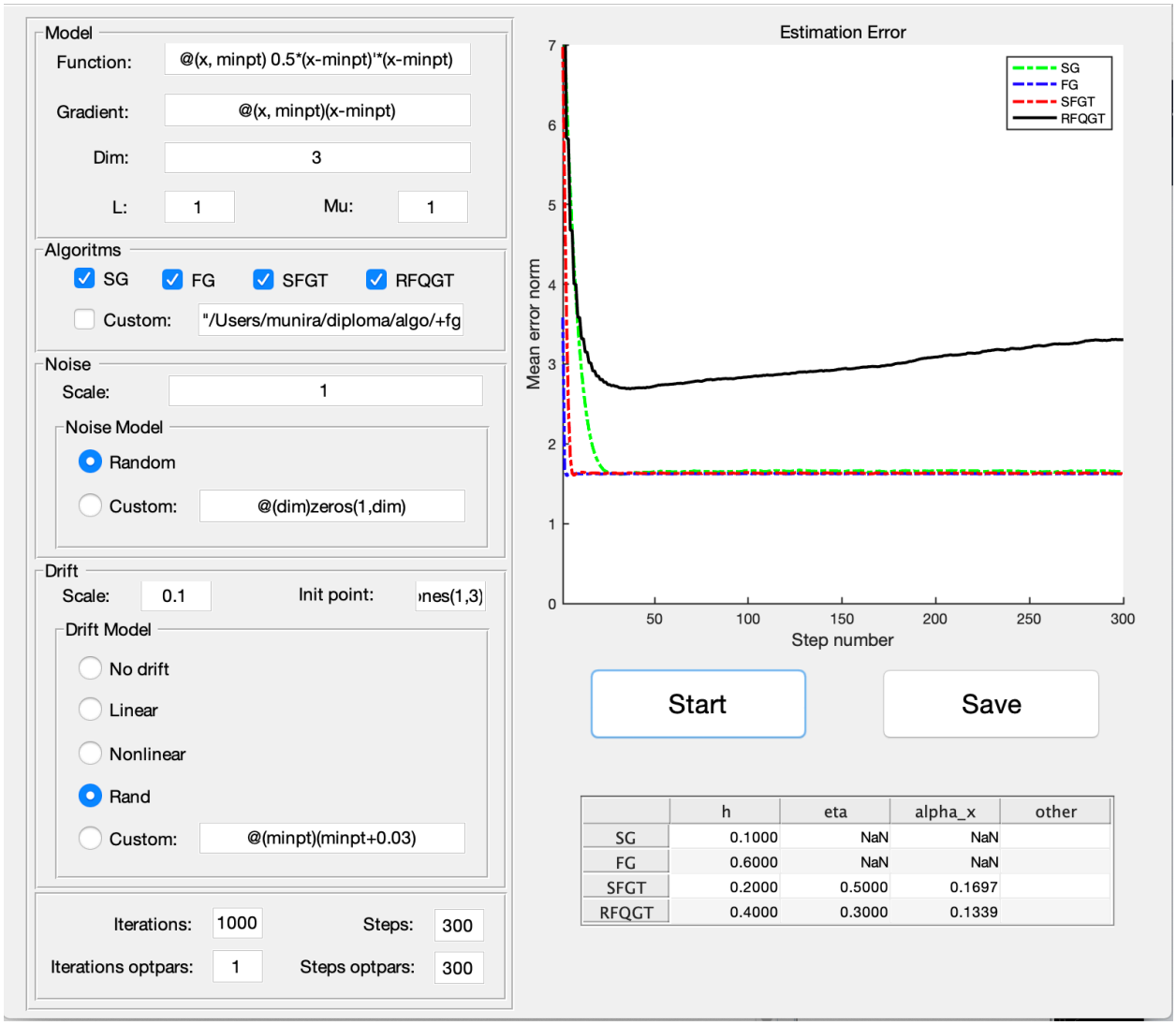


Рис. 12: Модель измерений размерности 3

Во-вторых, новый алгоритм можно применять с различными значениями шума, что делает его наиболее эффективным в моделях, с сильным зашумлением данных.

В будущем также планируется улучшить новый алгоритм RFQGT, путем изменения подхода к вычислению параметра $\Delta_n$ алгоритма, с целью улучшить оценку алгоритма на больших размерностях модели.

**Нагрузочное тестирование.** В ходе нагрузочного тестирования системы было проведено сравнение скорости работы всех алгоритмов, реализованных в системе. Тестирование системы проводилось на компьютере со следующими характеристиками: процессор Intel Core i5 с так-

| № | Шаги | Итер. | Шаги пар. | Итер. пар. | SG | FG | SFGT | RFQGT |
|---|---|---|---|---|---|---|---|---|
| 1 | 10 | 100 | 10 | 1 | 0.0288 | 0.0448 | 0.2439 | 0.2588 |
| 2 | 10 | 1000 | 10 | 1 | 0.3170 | 0.4422 | 0.6119 | 0.6070 |
| 3 | 10 | 1000 | 10 | 10 | 0.3326 | 0.4756 | 1.6215 | 1.6504 |
| 4 | 300 | 10 | 300 | 1 | 0.0826 | 0.1886 | 3.4318 | 3.4913 |
| 5 | 300 | 100 | 300 | 1 | 0.4335 | 0.9207 | 4.2197 | 4.3525 |
| 6 | 300 | 100 | 300 | 10 | 0.7497 | 1.5217 | 33.1889 | 35.7034 |
| 7 | 300 | 1000 | 300 | 1 | 3.8355 | 8.5027 | 12.9646 | 14.4403 |

Таблица 2: Среднее время работы алгоритмов в секундах

товой частотой 2.3GHz, ОЗУ 16 ГБ.

Для каждой серии экспериментов, приведенных в Таблице 2, среднее время работы каждого из алгоритмов расчитывалось по следующей формуле:

$$\mathbb{E} = \frac{1}{n} * \sum_{i=1}^{n} T_i,$$

где $n$ — количество экспериментов в серии, $T_i$ — время работы алгоритма в $i$-м эксперименте.

Как видно из Табицы 2, алгоритмы SFGT и RFQGT в среднем работают дольше алгоритмов SG и FG, поскольку в этих алгоритмах больше параметров, для которых вычисляется оптимальное значение. Также из Таблицы 2 можно сделать вывод, что в среднем время работы алгоритмов SFGT и RFQGT одинаково, однако, поскольку алгоритм RFQGT работает без знания знечения градиента и вычисление значения градиента порой требует существенных временных затрат, новый алгоритм значительно выигрывает по скорости, в моделях, в которых не задан градиент функции.

# Заключение

В ходе работы были достигнуты следующие результаты:

- Исследованы существующие алгоритмы оптимизации, которые можно применить в задаче трекинга.
- Разработан новый алгоритм нестационарной стохастической оптимизации «Рандомизированный быстрый квази-градиентный метод для задач трекинга» и исследованы его свойства.
- Исследованы существующие системы для моделирования алгоритмов оптимизации.
- Составлены требования к системе.
- Разработана архитектура системы.
- Реализованы классические методы стохастической оптимизации и новый метод «Рандомизированный быстрый квази-градиентный метод для задач трекинга» в системе.
- Разработан прототип системы, включающий в себя возможность визуализации невязок алгоритмов.
- Сделан графический интерфейс в системе.
- Проведена апробация системы.
- Сделан доклад на XXII конференции молодых ученых «Навигация и управление движением»

# Список литературы


[1] ALGLIB Documentation. — URL: https://www.alglib.net/docs.php.

[2] Blum R. G. Multidimensional stochastic approximation methods. — The Annals of Mathematical Statistics, 1954. — URL: https://www.researchgate.net/publication/38367267_Multidimensional_Stochastic_Approximation_Methods.

[3] Borkar V. S. Stochastic approximation: A dynamical systems viewpoint. — Cambridge University Press, 2008. — URL: https://www.springer.com/gp/book/9789386279385.

[4] Delyon B., Juditsky A. Asymptotical study of parameter tracking algorithms. — SIAM Journal on Control and Optimizationl, 1995. — URL: https://epubs.siam.org/doi/abs/10.1137/S0363012992238953?journalCode=sjcodc.

[5] Dual memory architectures for fast deep learning of stream data via an online-incremental-transfer strategy / S. W. Lee, M. O. Heo, J. Kim, B. T. Zhang. — 2015. — URL: https://arxiv.org/abs/1506.04477.

[6] Eweda E., Macchi O. Tracking error bounds of adaptive nonsta- tionary filtering. — Automatica, vol. 21, no. 3, 1985. — URL: https://www.semanticscholar.org/paper/Tracking-error-bounds-of-adaptive-nonstationary-Eweda-Macchi/009d3d405ea305831b1397dbaf47047406891a61.

[7] Ganegedara T., Ott L., Ramos F. Online adaptation of deep architectures with reinforcement learning. — Proceedings of the 22nd European Conference on Artificial Intelligence, 2016. — URL: https://www.researchgate.net/publication/305994967_Online_Adaptation_of_Deep_Architectures_with_Reinforcement_Learning.

[8] Gekko. — URL: https://gekko.readthedocs.io/en/latest/.

[9] Gould N. I. M., Orban D., Toint P. L. CUTEr and SifDec: A constrained and unconstrained testing environment, revisited. — ACM Transactions on Mathematical Software, 2003.

[10] Granichin O., Amelina N. Simultaneous perturbation stochastic approximation for tracking under unknown but bounded disturbances // IEEE Transactions on Automatic Control. — 2015. — URL: https://ieeexplore.ieee.org/document/6908991.

[11] Granichin O, Gurevich L., Vakhitov A. Discrete-time minimum tracking based on stochastic approximation algorithm with randomized differences // Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference. — 2009. — URL: https://ieeexplore.ieee.org/document/5400839.

[12] J. Kiefer J. Wolfowitz. Stochastic estimation of the maximum of a regression function. — The Annals of Mathematical Statistics, 1952. — URL: https://projecteuclid.org/euclid.aoms/1177729392.

[13] Kingma D., Ba J. Adam: A method for stochastic optimization. — International Conference on Learning Representations (ICLR), 2014. — URL: https://www.researchgate.net/publication/269935079_Adam_A_Method_for_Stochastic_Optimization.

[14] Kushner H. J., Huang H. Asymptotic properties of stochastic approximations with constant coefficients. — SIAM Journal on Control and Optimization, vol. 19, no. 1, 1981. — URL: https://www.researchgate.net/publication/243092849_Asymptotic_Properties_of_Stochastic_Approximations_with_Constant_Coefficients.

[15] Kushner H. J., Yin G. G. Stochastic Approximation and Recursive Algorithms and Applications. — Springer Science and Business Media, 2003. — URL: https://link.springer.com/book/10.1007/b97441.

[16] MOSEC Documentation. — URL: https://www.mosek.com/documentation/.

[17] Mixed Integer Distributed Ant Colony Optimization. — URL: http://www.midaco-solver.com/index.php/about.

[18] Nesterov Y.E. A method of solving a convex programming problem with convergence rate o(1/k2). — Soviet Mathematics Doklady, 1983. — URL: https://ci.nii.ac.jp/naid/10029946121/.

[19] Nesterov Y.E. Introductory Lectures on Convex Optimization: A Basic Course. — Springer Science and Business Media, 2013. — URL: https://www.springer.com/gp/book/9781402075537.

[20] On the importance of initialization and momentum in deep learning / I. Sutskever, J. Martens, G. E. Dahl, G. E. Hinton. — Proceedings of the 30th International Conference on International Conference on Machine Learning, 2013. — URL: https://dl.acm.org/doi/10.5555/3042817.3043064.

[21] Optimization Toolbox Documentation - MathWorks. — URL: https://www.mathworks.com/help/optim/.

[22] Polyak B. T. Some methods of speeding up the convergence of iteration methods. — USSR Computational Mathematics and Mathematical Physics, 1964. — URL: https://www.sciencedirect.com/science/article/abs/pii/0041555364901375?via.

[23] Polyak B. T. Introduction to Optimization. — Optimization Software, 1987. — URL: https://www.researchgate.net/publication/268248877_Introduction_to_optimization_Vvedenie_v_optimizatsiyu.

[24] Popkov A. Y. Gradient methods for nonstationary unconstrained optimization problems. — Automation and Remote Control, 2005. — URL: https://link.springer.com/article/10.1007/s10513-005-0132-z.

[25] Robbins H., Monro S. A stochastic approximation method. — The annals of mathematical statistics, 1951. — URL: https://projecteuclid.org/euclid.aoms/1177729586.

[26] SciPy. — URL: https://www.scipy.org/about.html.

[27] Spall J. C. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. — IEEE Transactions on Automatic Control, 1992. — URL: https://www.researchgate.net/publication/3021008_Multivariate_stochastic_approximation_using_a_simultaneous_perturbation_gradient_approximation.

[28] Stochastic approximation: A dynamical systems viewpoint / C. Tessler, S. Givony, T. Zahavy et al. — 2008. — URL: https://www.springer.com/gp/book/9789386279385.

[29] Stochastic fast gradient for tracking / O Granichin, A. Vakhitov, D. Kosaty, M. Yuchi // American Control Conference (ACC). — 2019. — URL: https://ieeexplore.ieee.org/document/8815070.

[30] TOMLAB Optimization. — URL: https://tomopt.com/tomlab/about/.

[31] A deep hierarchical approach to lifelong learning in minecraft / C. Tessler, S. Givony, T. Zahavy et al. — Proceedings of the 31th Conference on Artificial Intelligence, 2017. — URL: https://arxiv.org/abs/1604.07255.

[32] Руководство по GAMS / А. Брук, Д. Кендрик, А Meeраус, Р Раман. — 1999. — URL: https://www.gams.com/fileadmin/community/contrib/doc/gamsman_russian.pdf.