Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 7**

Cравнительный анализ эффективности численных методов нулевого порядка для поиска безусловного экстремума

Тема

Преподаватель В. В. Тынченко

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Подпись, дата Инициалы, Фамилия

Студент КИ19-17/1Б, №031939174 А. К. Никитин

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Номер группы, зачетной книжки Подпись, дата Инициалы, Фамилия

Красноярск 2021

# Постановка задачи

На основании результатов выполнения практических работ модуля "Численные методы нулевого порядка для поиска безусловного экстремума" сравнить реализованные алгоритмы по точности и скорости решения задач оптимизации, варьируя параметры алгоритмов. Для проведения вычислительных экспериментов самостоятельно выбрать 3 целевые функции и интервалы неопределенности, интересные с точки зрения исследования. Результаты вычислительных экспериментов представить в табличном виде, прокомментировать их и сделать обоснованный вывод об особенностях работы исследуемых алгоритмов и их эффективности на различных целевых функциях.

# Функции для исследования

## Функции одной переменной

Ниже представлен список функций одной переменной.

1. f(x) = x4 + 2x2 + 4x + 1
2. f(x) = x \* (x – 1)2 \* (x – 3)3
3. f(x) = ex – 2x + 4(x + 2)2

## Функции нескольких переменных

Ниже представлен список функций нескольких переменных.

1. f(x1, x2) = 3x12 + x1 x2 + 2x22 - x1 - 4 x2
2. f(x1, x2) = (x1 - 4)2 + (x2 - 1)2
3. f(x1, x2) = 2x12 - x2 + 18x22

# Исходные тексты программ

На листинге 1, 2 представлен код программы, проводящий анализ функций.

Листинг 1 – Анализ функций одной переменной

from fibonacci import fibonacci\_method as fibonacci

from powell import powell\_method\_1 as powell

from matplotlib import pyplot as plt

from const import Function

from sympy import Symbol

import numpy as np

import math

from time import time

x = Symbol('x')

param\_start = 0.01

param\_end = 2.

param\_step = 0.02

function\_interval = (-20, 20)

plt.rcParams["figure.figsize"] = (10, 5)

def assess(method, \*args, \*\*kwargs):

t1 = time()

extremum = method(\*args, \*\*kwargs)[0]

t2 = time()

return extremum, t2 - t1

def compare(func, exact\_extremum):

deltas = {'powell': [], 'fib': []}

iterations = {'powell': [], 'fib': []}

epsilons = []

for param in np.arange(param\_start, param\_end, param\_step):

powell\_results = assess(powell, func, function\_interval, epsilon1=param, epsilon2=param)

fibonacci\_results = assess(fibonacci, func, function\_interval, l=param)

Окончание листинга 1

deltas['powell'].append(abs(powell\_results[0] - exact\_extremum))

deltas['fib'].append(abs(fibonacci\_results[0] - exact\_extremum))

iterations['powell'].append(powell\_results[1])

iterations['fib'].append(fibonacci\_results[1])

epsilons.append(param)

fig, ax = plt.subplots(nrows=1, ncols=2)

ax[0].plot(epsilons, deltas['powell'], c=np.random.rand(3, ))

ax[0].plot(epsilons, deltas['fib'], c=np.random.rand(3, ))

ax[0].set\_xlabel('epsilon, l')

ax[0].set\_ylabel('accuracy')

ax[0].legend(['Пауэлл', 'Фибоначчи'])

ax[1].plot(epsilons, iterations['powell'], c=np.random.rand(3, ))

ax[1].plot(epsilons, iterations['fib'], c=np.random.rand(3, ))

ax[1].set\_xlabel('epsilon, l')

ax[1].set\_ylabel('iterations\_number')

ax[1].legend(['Пауэлл', 'Фибоначчи'])

plt.show()

if \_\_name\_\_ == '\_\_main\_\_':

func1 = Function(x \*\* 4 + 2 \* x \*\* 2 + 4 \* x + 1, x)

func2 = Function(x \* (x - 1)\*\*2 \* (x - 3)\*\*3, x)

func3 = Function(math.e\*\*x - 2 \* x + 4 \* (x + 2)\*\*2, x)

exact\_extremum1 = -0.682

exact\_extremum2 = 0.263

exact\_extremum3 = -1.77

compare(func3, exact\_extremum3)

Листинг 2 – Анализ функций нескольких переменных

from hooke\_jeeves import hooke\_jeeves\_method

from nelder\_mead import nelder\_mead\_algorithm

from powell import powell\_method\_n

from rosenbrock import rosenbrock\_algorithm

from matplotlib import pyplot as plt

from const import Function

from sympy import Symbol

import numpy as np

import math

Продолжение листинга 2

from time import time

x1 = Symbol('x1')

x2 = Symbol('x2')

param\_start = 0.01

param\_end = 2.

param\_step = 0.02

function\_interval = [-50, 50]

x0 = [10, -10]

plt.rcParams["figure.figsize"] = (10, 5)

def assess(method, \*args, \*\*kwargs):

t1 = time()

extremum = method(\*args, \*\*kwargs)[0]

t2 = time()

return extremum, t2 - t1

def compare(func, exact\_extremum):

deltas = {'powell': [], 'hooke': [], 'nelder': [], 'rosenbrock': []}

iterations = {'powell': [], 'hooke': [], 'nelder': [], 'rosenbrock': []}

epsilons = []

for param in np.arange(param\_start, param\_end, param\_step):

hooke\_jeeves\_results = assess(hooke\_jeeves\_method, func, x0, epsilon=param, steps=[2.01, 2.01])

nelder\_mead\_results = assess(nelder\_mead\_algorithm, func, ((-10, -10), (10, -10), (0, 10)), epsilon=param)

rosenbrock\_results = assess(rosenbrock\_algorithm, func, x0, epsilon=param)

powell\_results = assess(powell\_method\_n, func, x0, function\_interval, epsilon=param)

delta = sum([abs(exact\_extremum[i] - powell\_results[0][i]) for i in range(len(exact\_extremum))])

deltas['powell'].append(delta)

delta = sum([abs(exact\_extremum[i] - hooke\_jeeves\_results[0][i]) for i in range(len(exact\_extremum))])

Продолжение листинга 2

deltas['hooke'].append(delta)

delta = sum([abs(exact\_extremum[i] - nelder\_mead\_results[0][i]) for i in range(len(exact\_extremum))])

deltas['nelder'].append(delta)

delta = sum([abs(exact\_extremum[i] - rosenbrock\_results[0][i]) for i in range(len(exact\_extremum))])

deltas['rosenbrock'].append(delta)

iterations['powell'].append(powell\_results[1])

iterations['hooke'].append(hooke\_jeeves\_results[1])

iterations['nelder'].append(nelder\_mead\_results[1])

iterations['rosenbrock'].append(rosenbrock\_results[1])

epsilons.append(param)

fig, ax = plt.subplots(nrows=1, ncols=2)

powell\_color = np.random.rand(3, )

ax[0].plot(epsilons, deltas['powell'], c=powell\_color)

hooke\_color = np.random.rand(3, )

ax[0].plot(epsilons, deltas['hooke'], c=hooke\_color)

nelder\_color = np.random.rand(3, )

ax[0].plot(epsilons, deltas['nelder'], c=nelder\_color)

rosenbrock\_color = np.random.rand(3, )

ax[0].plot(epsilons, deltas['rosenbrock'], c=rosenbrock\_color)

ax[0].set\_xlabel('epsilon')

ax[0].set\_ylabel('accuracy')

ax[0].legend(['Пауэлл', 'Хук-Дживс', 'Нелдер-Мид', 'Розенброк'])

ax[1].plot(epsilons, iterations['powell'], c=powell\_color)

ax[1].plot(epsilons, iterations['hooke'], c=hooke\_color)

ax[1].plot(epsilons, iterations['nelder'], c=nelder\_color)

ax[1].plot(epsilons, iterations['rosenbrock'], c=rosenbrock\_color)

ax[1].set\_xlabel('epsilon')

ax[1].set\_ylabel('iterations\_number')

ax[1].legend(['Пауэлл', 'Хук-Дживс', 'Нелдер-Мид', 'Розенброк'])

plt.show()

Окончание листинга 2

if \_\_name\_\_ == '\_\_main\_\_':

func1 = Function(3 \* x1\*\*2 + x1 \* x2 + 2 \* x2\*\*2 - x1 - 4 \* x2, (x1, x2))

func2 = Function((x1 - 4)\*\*2 + (x2 - 1)\*\*2, (x1, x2))

func3 = Function(2 \* x1\*\*2 - x2 + 18 \* x2\*\*2, (x1, x2))

exact\_extremum1 = [0, 1]

exact\_extremum2 = [4, 1]

exact\_extremum3 = [0, 1/36]

compare(func3, exact\_extremum3)

# Сравнительный анализ методов

## Методы функций одной переменной

На рисунках 1, 2, 3 представлены графики изменения точности и производительности методов Фибоначчи и Пауэлла от параметра epsilon.

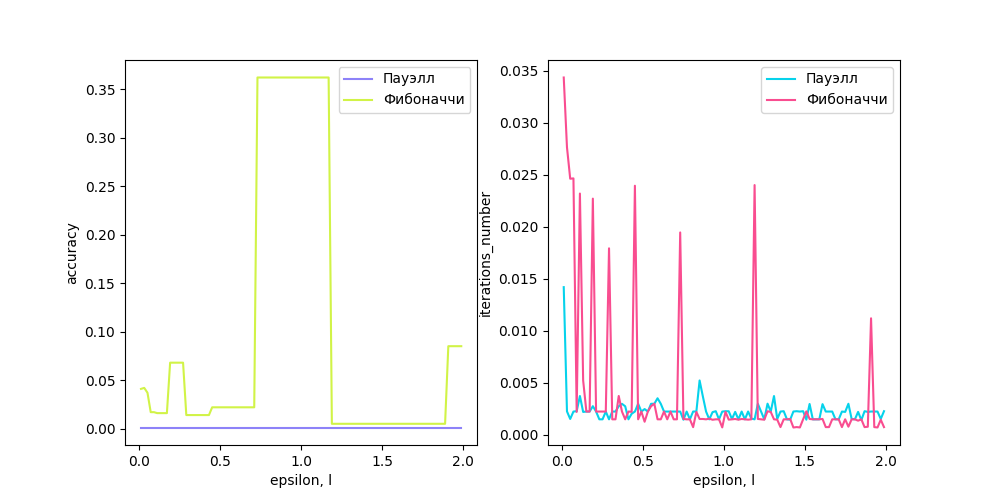


Рисунок 1 – Изменение точности алгоритмов от параметра epsilon функции f(x) = x4 + 2x2 + 4x + 1

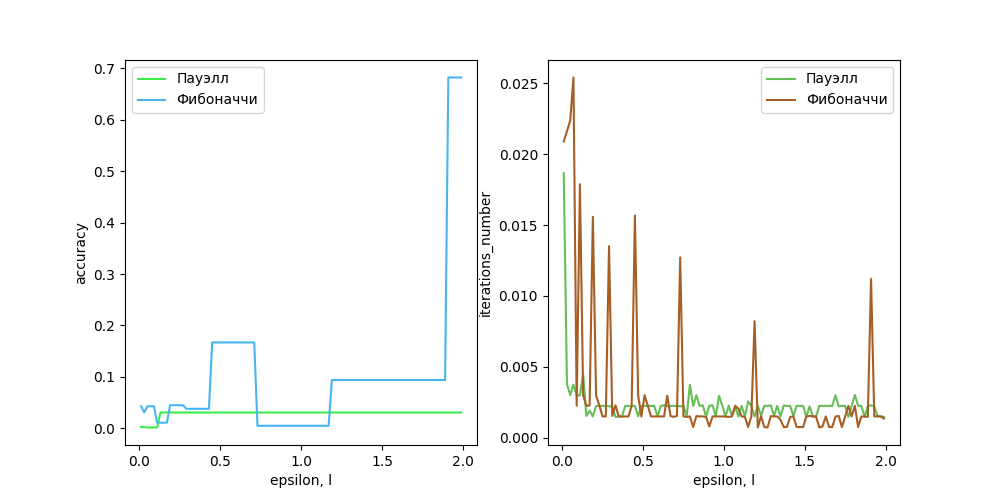


Рисунок 2 – Изменение точности алгоритмов от параметра epsilon функции f(x) = x \* (x – 1)2 \* (x – 3)3

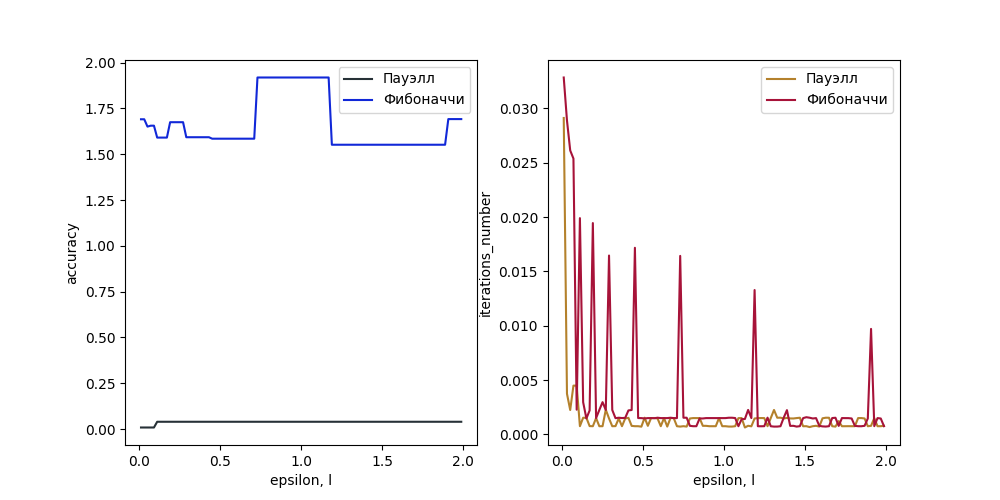


Рисунок 3 – Изменение точности алгоритмов от параметра epsilon функции f(x) = ex – 2x + 4(x + 2)2

## Методы функций нескольких переменных

На рисунках 4, 5, 6 представлены графики изменения точности и производительности методов Хука-Дживса, Пауэлла, Розенброка, Нелдера-Мида от параметра epsilon.

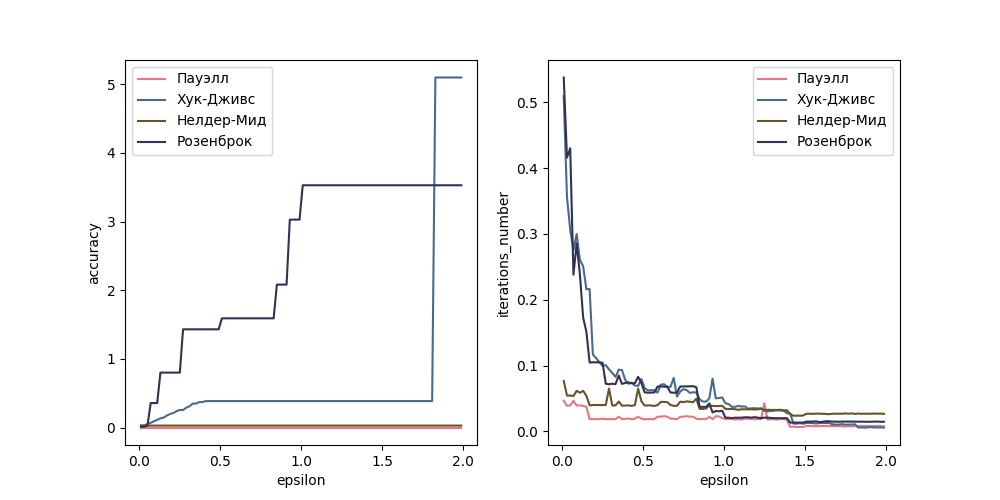


Рисунок 4 – Изменение точности алгоритмов от параметра epsilon функции f(x1, x2) = 3x12 + x1 x2 + 2x22 - x1 - 4 x2

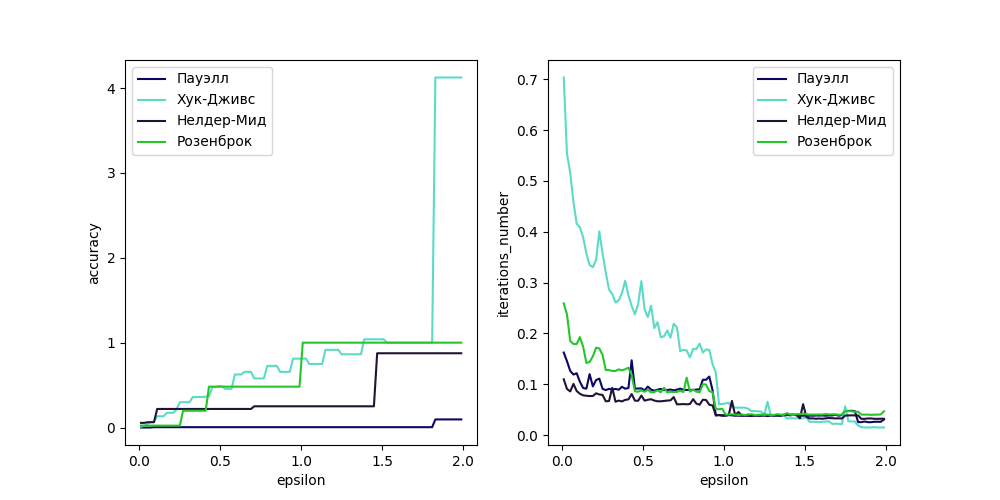


Рисунок 5 – Изменение точности алгоритмов от параметра epsilon функции f(x1, x2) = (x1 - 4)2 + (x2 - 1)2

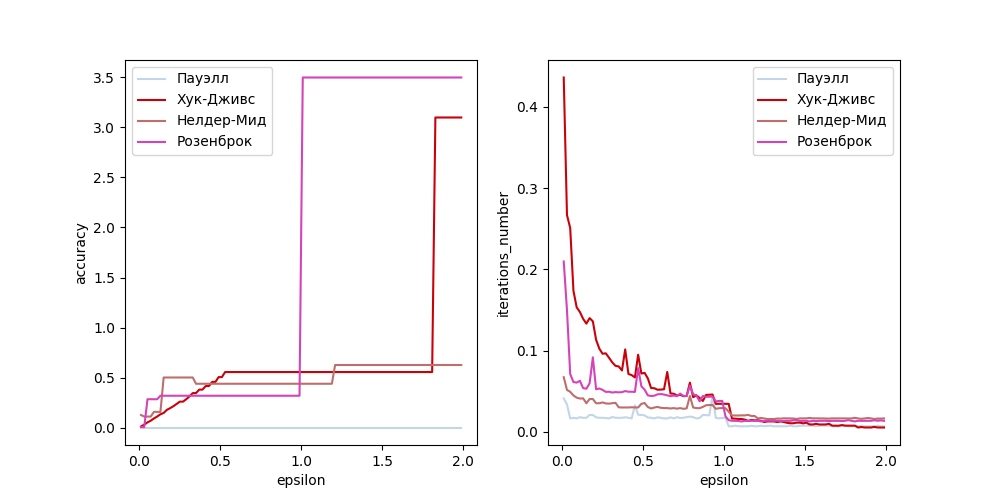


Рисунок 6 – Изменение точности алгоритмов от параметра epsilon функции f(x1, x2) = 2x12 - x2 + 18x22

# Вывод

Таким образом, для минимизации функций одной переменной наиболее эффективным оказался метод Пауэлла; для минимизации функций нескольких переменных переменной наиболее эффективным оказался метод сопряженных направлений.