МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение   
высшего образования   
«Санкт-Петербургский государственный технологический институт  
(технический университет)»  
СПбГТИ(ТУ)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| УГНС | | 09.00.00 | Информатика и вычислительная техника | | |
| Направление подготовки | | 09.03.01 | Информатика и вычислительная техника | | |
| Направленность (профиль) | |  | Системы автоматизированного проектирования | | |
| Форма обучения | |  | очная | | |
|  | |  |  | | |
| Факультет | |  | Информационных технологий и управления | | |
| Кафедра | |  | Систем автоматизированного  проектирования и управления | | |
| Учебная дисциплина | |  | Методы оптимизации | | |
| Курс | 3 | | | Группа | 454 |

КУРСОВОЙ ПРОЕКТ

|  |  |
| --- | --- |
| Тема: | Разработка программного комплекса для решения задачи оптимизации заданного объекта исследования |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент |  |  |  | Дамрин А.О. |
| Руководитель |  |  |  | Смирнов И.А. |
| Оценка за курсовой проект |  |  |  |  |

Санкт-Петербург

2018

**Содержание**

[Введение 3](#_Toc518595519)

[1 Аналитический обзор 4](#_Toc518595520)

[1.1 Роль и место оптимизационных задач в автоматизированном проектировании химических производств 4](#_Toc518595521)

[1.2 Обоснование выбора метода оптимизации для решения поставленной задачи 5](#_Toc518595522)

[1.3 Обоснование выбора инструментальных средств разработки программного комплекса 6](#_Toc518595523)

[2 Цели и задачи 7](#_Toc518595524)

[3 Технологическая часть 8](#_Toc518595525)

[3.1 Формализованное описание объекта оптимизации 8](#_Toc518595526)

[3.2 Разработка блок-схемы алгоритма функционирования программного комплекса 10](#_Toc518595527)

[3.3 Разработка структуры пользовательского интерфейса. UML-диаграммы вариантов использования для администратора и пользователя 11](#_Toc518595528)

[3.4 Разработка программной реализации алгоритма решения задачи 12](#_Toc518595529)

[3.5 Разработка пользовательского интерфейса 14](#_Toc518595530)

[Заключение 19](#_Toc518595531)

[Список используемой источников 20](#_Toc518595532)

[Приложение А. Листинг программы 21](#_Toc518595533)

# **Введение**

Методы оптимизации в автоматизированных системах являются дисциплиной, предусмотренной рабочим планом специальности 09.03.01 “Информатика и вычислительная техника”.

Целью курсового проекта является закрепление практических навыков, полученных на лабораторных занятиях, и получение новых навыков по решению оптимизационных задач с использованием методов оптимизации в ходе выполнения курсового проекта. Также необходимо создать программный комплекс, который позволит решать различные задачи оптимизации и отображать их решения в графическом виде, то есть построение двумерных и трехмерных графиков.

В ходе выполнения курсового проекта на необходимо формализовать объект и задачу оптимизации, выбрать оптимальный метод решения, представить решение задачи в текстовом и графическом виде, разработка библиотеки оптимизационных задач и методов оптимизации.

Задачей курсового проекта является получение программного комплекса, который сможет решить различные оптимизационные задачи.

1. **Аналитический обзор**
   1. **Роль и место оптимизационных задач в автоматизированном проектировании химических производств**

Оптимизация предполагает достижения наилучших или определение наиболее благоприятных условий проведения химического процесса.

При оптимизации химико-технологических систем (ХТС) качество функционирования систем определяют с помощью критериев, или показателей эффективности, под которыми понимают числовые характеристики системы, оценивающие степень приспособления системы к выполнению поставленных перед ней задач.

Основные показатели эффективности ХТС следующие:

- технологические показатели;

- качественные показатели выпускаемой продукции;

- экономические показатели.

Показателями эффективности отдельных аппаратов, узлов, отделений чаще всего являются технологические критерии – удельная производительность, выход целевого продукта, КПД и т. д. Технологическими критериями эффективности являются степень конверсии, избирательность (доля превращенного исходного реагента, израсходованная на образование данного продукта), выход продукта. Различают две стадии оптимизации: статическую и динамическую. В зависимости от характера рассматриваемых математических моделей применяются различные математические методы оптимизации: аналитические, методы математического программирования, градиентные и статические.

Задача оптимизации при проектировании химического производства может заключаться в решении следующих проблем:

* Как спроектировать и наладить химическое производство с минимальными затратами времени, человеческих и финансовых ресурсов;
* Какие агрегаты выбрать для проведения химических реакций;
* Какое сырье использовать в ходе проведения химических реакций;
* Как обеспечить доставку исходного сырья на завод и отправку готового продукта потребителям.

Оптимальное решение этих и многих других вопросов и является основной целью оптимизации при проектировании химических производств.

Также не следует забывать и о том, что оптимизационные задачи ставятся и решаются не только при проектировании химического производства, но и в ходе его функционирования.

Оптимизационные задачи в химическом производстве формулируются и решаются на протяжении всего времени: во время моделирования, создания и функционирования. Поэтому роль оптимизационных задач очень большая и без процесса оптимизации на данный момент невозможно представить ни одно химическое производство.

## 1.2 Обоснование выбора метода оптимизации для решения поставленной задачи

Правильный выбор метода оптимизации чрезвычайно важен для корректного и быстрого решения поставленной оптимизационной задачи. Теорию и методы решения задачи оптимизации изучает математическое программирование. Математическое программирование — это область математики, разрабатывающая теорию, численные методы решения многомерных задач с ограничениями. В отличие от классической математики, математическое программирование занимается математическими методами решения задач нахождения наилучших вариантов из всех возможных.

Общая запись задач оптимизации задаёт большое разнообразие их классов. От класса задачи зависит подбор метода (эффективность её решения). Классификацию задач определяют: целевая функция и допустимая область (задаётся системой неравенств и равенств или более сложным алгоритмом). Методы оптимизации классифицируют в соответствии с задачами оптимизации:

* Локальные методы: сходятся к какому-нибудь локальному экстремуму целевой функции. В случае унимодальной целевой функции, этот экстремум единственен, и будет глобальным максимумом/минимумом.
* Глобальные методы: имеют дело с многоэкстремальными целевыми функциями. При глобальном поиске основной задачей является выявление тенденций глобального поведения целевой функции.

Существующие в настоящее время методы поиска можно разбить на три большие группы:

* детерминированные;
* случайные (стохастические);
* комбинированные.

По критерию размерности допустимого множества, методы оптимизации делят на методы одномерной оптимизации и методы многомерной оптимизации. По виду целевой функции и допустимого множества, задачи оптимизации и методы их решения можно разделить на Задачи дискретного, целочисленного, нелинейного и линейного программирования (по виду ограничений и целевой функции). Помимо того, оптимизационные методы делятся на следующие группы: аналитические, численные и графические методы оптимизации.

Исходя из поставленной задачи, вида целевой функции и ограничений, можно охарактеризовать поставленную задачу с точки зрения математического программирования и выбрать подходящий для ее решения набор методов, пользуясь вышеприведенной классификацией. В задаче необходимо найти глобальный условный экстремум, целевая функция не линейна, поэтому выбираем метод нелинейного программирования, задача оптимизации двумерная, так как целевая функция зависит от двух переменных и выбираем детерминированный численный метод оптимизации.

Итак, для решения поставленной задачи оптимизации был выбран метод полного перебора с постоянным шагом (метод сканирования), поскольку он является самым простым для понимания и реализации методом. Выбор данного метода также обусловлен тем, что целевая функция достаточно проста для вычисления, а число ее аргументов ограничено 2-мя, что позволяет легко перебрать все возможные их значения.

## 1.3 Обоснование выбора инструментальных средств разработки программного комплекса

Для выполнения поставленной задачи была выбрана кроссплатформенная свободная IDE Qt Creator для работы с фреймворком Qt на языке С++.

Qt-кроссплатформенный фреймворк для разработки программного обеспечения на языке программирования С++. Qt позволяет запускать написанное с его помощью программное обеспечение в большинстве современных операционных систем путем простой компиляции программы для каждой системы без изменения исходного кода. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью, базами данных и XML. Является полностью объектно-ориентированным, расширяемым и поддерживающим технику компонентного программирования.

Основная задача Qt Creator – упростить разработку приложений с помощью фреймворка Qt на разных платформах. Поэтому среди возможностей, присущих любой среде разработки, есть и специфичные, такие ка отладка приложений на QML и отображение в отладчике данных из контейнеров Qt, встроенный дизайнер интерфейсов как на QML, так и на QtWidgets.

Мною была использована последняя версия - Qt Creator 4.60 и Qt 5.10.

# **2 Цели и задачи**

Целью курсового проекта является закрепление практических навыков, полученных на лабораторных занятиях, и получение новых навыков по решению оптимизационных задач с использованием методов оптимизации в ходе выполнения курсового проекта. Также необходимо создать программный комплекс, который позволит решать различные задачи оптимизации и отображать их решения в графическом виде, то есть построение двумерных и трехмерных графиков.

Задачей курсового проекта является получение программного комплекса, который сможет решить различные оптимизационные задачи.

В ходе выполнения курсового проекта на необходимо формализовать объект и задачу оптимизации, выбрать оптимальный метод решения, представить решение задачи в текстовом и графическом виде, разработка библиотеки оптимизационных задач и методов оптимизации.

# **3 Технологическая часть**

## 3.1 Формализованное описание объекта оптимизации

Основными этапами формализации:

1 Формирование системы неизвестных. Выявление элементов, описывающих структуру моделируемой системы, и описание их в виде переменных;

2 Формирование системы ограничений. Описание в формальном виде условий, которые должны быть соблюдены при реализации задач;

3 Формулирование критерия оптимальности и запись его в виде целевой функции задачи математического программирования.

Итак, имеется текст исходной задачи оптимизации:

*Объектом оптимизации является химический реактор, в котором происходит образование целевого компонента. Реактор оборудован мешалкой и двумя теплообменными устройствами: змеевиком и рубашкой. Необходимо определить температурные условия технологического процесса, обеспечивающие максимальную прибыль от реализации целевого компонента. Согласно эмпирической математической модели, количество получаемого целевого компонента P (кг) связано с параметрами процесса следующим образом:*

*где α, β, γ – нормирующие множители, равные 1;*

*G – количество реакционной массы (1 кг);*

*H – давление в реакторе (9 Кпа);*

*N – скорость вращения мешалки (10 об/с);*

*Т1, Т2 – температуры в теплообменных устройствах (°С).*

*Регламентом установлено, что температура в змеевике может изменяться в диапазоне от -2 до 15 °С, в рубашке – от -2 до 12 °С. Кроме того, должно выполняться условие T1 + T2 ≤ 12 °С.*

*Прибыль от реализации 1 кг целевого компонента составляет 100 у.е. Точность решения – 10 у.е.*

Для решения задачи оптимизации необходимо найти точку глобального максимума целевой функции с учетом системы ограничений, наложенных на ее параметры. Это и будет являться формализованным описанием задачи оптимизации.

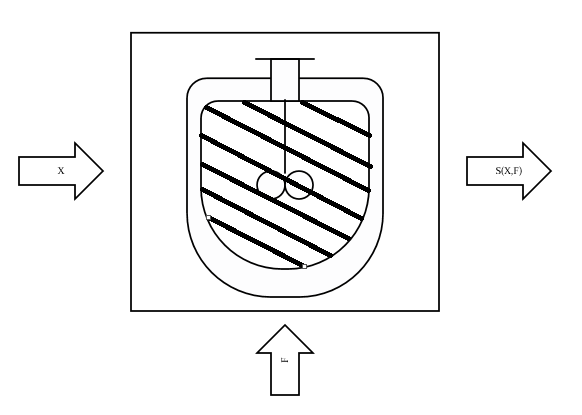
Формализованное описание объекта оптимизации представлено на рисунке 1.

Рисунок 1 – Формализованное описание объекта оптимизации

Условные обозначения, использованные при описании объекта оптимизации:

* X – вектор входных параметров;
* F – вектор управляющих воздействий;
* S (X, F) – целевая функция на выходе объекта, зависящая от входных параметров и управляющих воздействий;

Входными (т.е. неизменными) параметрами являются все нормирующие коэффициенты функции Р α = β= γ =1), количество реакционной массы (G = 1 кг), давление в реакторе (H = 9 Кпа), скорость вращения мешалки (N = 10 об/с). Управляющими воздействиями в таком случае будут являться расходы компонентов Т1 и Т2.

Целевой функцией в данном случае будет являться функция себестоимости S, зависящая от количества компонента (Р), и стоимости реализации 1 кг (С). Также, чтобы формально выразить целевую функцию и формализовать задачу, необходимо от обозначений Т1 и Т2 перейти к стандартным X1 и X2.

Тогда формализованная задача оптимизации будет ставиться следующим образом:

В соответствии с условиями задачи, необходимо найти максимум целевой функции S при заданной системе ограничений для ее параметров X1 и X2.

## 3.2 Разработка блок-схемы алгоритма функционирования программного комплекса

Программный комплекс предоставляет возможности:

* выбор оптимизационных задач
* выбор метода решения задач
* решить оптимизационные задачи
* выводить решения в различных представлениях и т.д.

Также программный комплекс предполагает наличие администратора, который будет осуществлять управление оптимизационными задачами и методами оптимизации, которые доступны пользователю.

На рисунке 2 изображена блок-схема алгоритма функционирования программного комплекса.



Рисунок 2 - Блок-схема алгоритма функционирования программного комплекса

## 3.3 Разработка структуры пользовательского интерфейса. UML-диаграммы вариантов использования для администратора и пользователя

В программном комплексе реализованы две категории пользователей: администратор и обычного пользователя. На рисунках 3 и 4 показаны UML-диаграммы вариантов использования для администратора и пользователя.



Рисунок 3 - UML-диаграммы вариантов использования для администратора

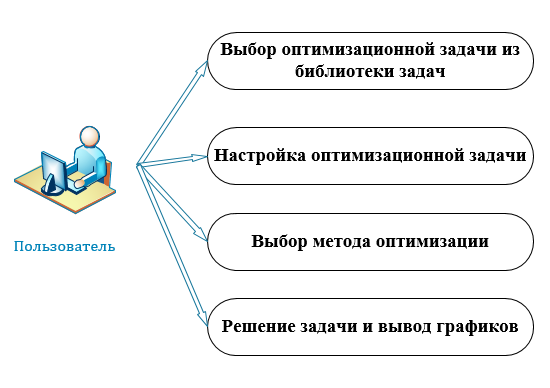


Рисунок 4 - UML-диаграммы вариантов использования для пользователя

## 3.4 Разработка программной реализации алгоритма решения задачи

Основным алгоритмом решения задачи оптимизации является метод сканирования или метод полного перебора. Обоснование выбора этого алгоритма описано в аналитической части. На рисунке 5 изображена блок-схема этого алгоритма.

X1min, X2min, X1max, X2max, par, parfunc, func, step

Начало

X1=X1min; X1≤ X1max; X1+=step

X2=X2min; X2≤ X2max; X2+=step

parfunc(X1, X2)≤par

func(X1,X2)>max

X1P=X1, X2P=X2, FMAX=func(X1,X2)

X1P, X2P, FMAX

Конец

нет

да

да

да

да

нет

нет

нет

Рисунок 5 – Блок-схема алгоритма решения задачи

## 3.5 Разработка пользовательского интерфейса

Схематическое изображения интерфейса программного комплекса изображено на рисунке 6.

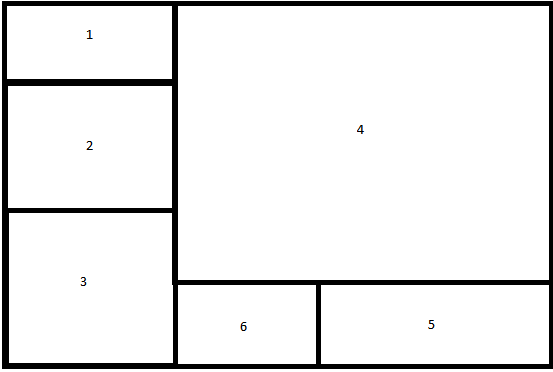


Рисунок 6 – Схематическое изображение интерфейса программного комплекса.

На рисунке 6 показано, что интерфейс программы разделен на 6 частей. Первая часть отвечает за выбор задачи и метода решения. Вторая часть отвечает за варьируемые переменные, в ней можно изменять ограничения температур змеевика и рубашки. В третьей части расположены не варьируемые переменные, которые нельзя изменять. В четвертой – показан 2D – график с линиями равного значения, в пятой – таблица с результатами вычислений, в шестой – находятся кнопки для расчета, построения 2D – графика и построения 3D- графика.

На рисунках 7, 8 и 9 показаны изображения интерфейса уже написанной программы.

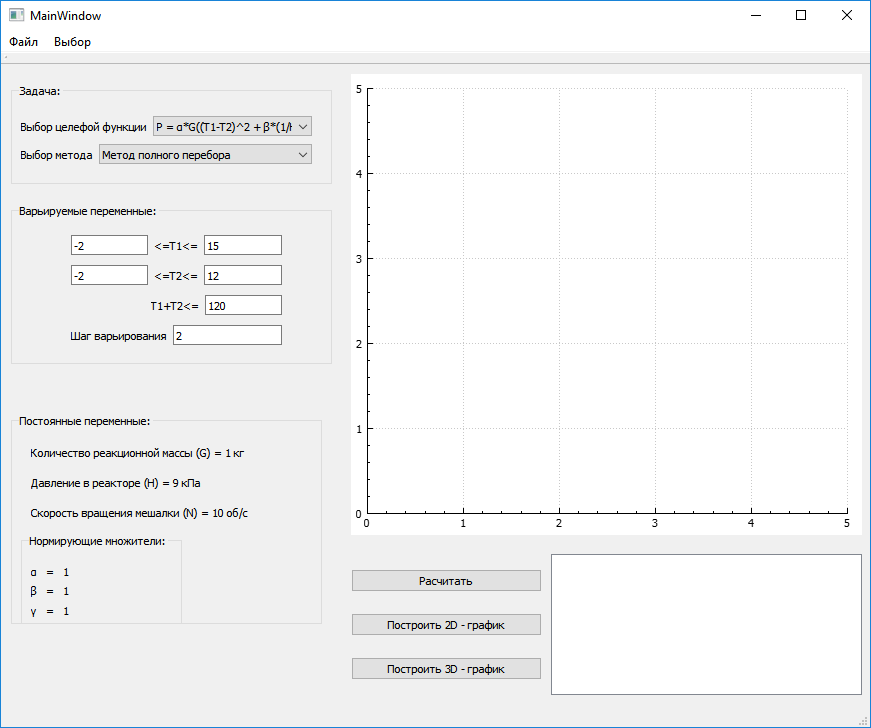


Рисунок 7 – Пользовательское окно

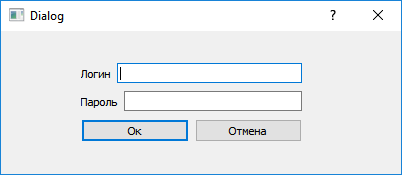


Рисунок 8 – Окно авторизации

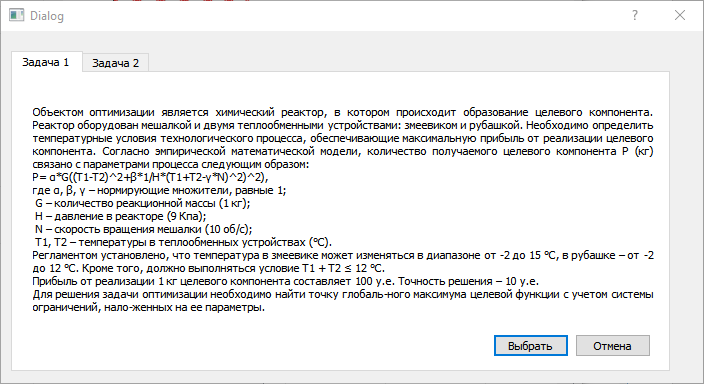


Рисунок 8 – Окно выбора задачи оптимизации

На рисунках 9, 10, 11 и 12 показаны результаты вычислений.

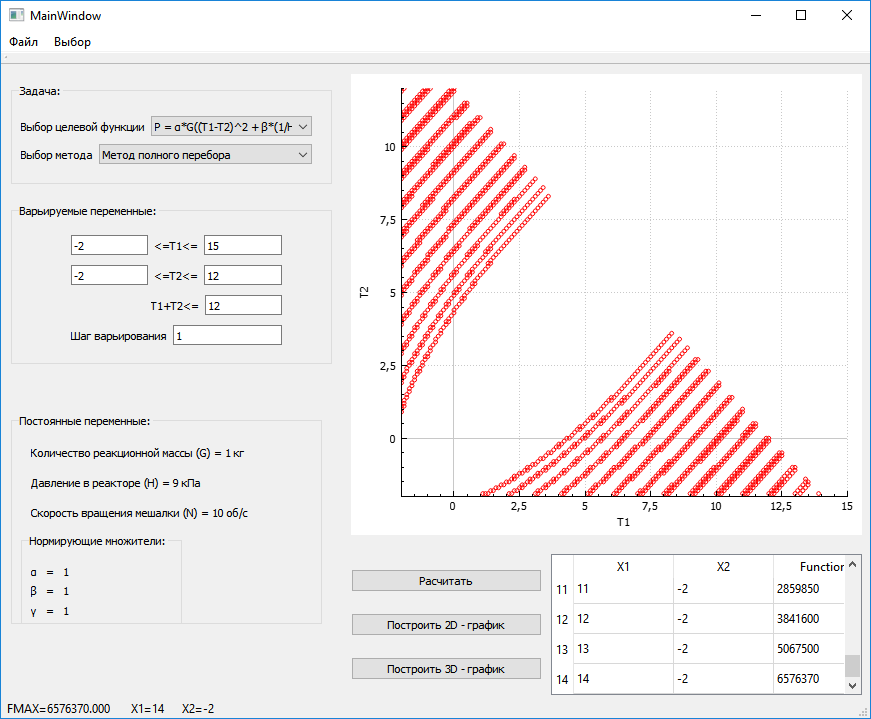


Рисунок 9 – Главное окно с результатами

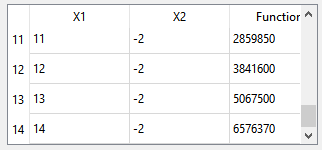


Рисунок 10 – Таблица результатов

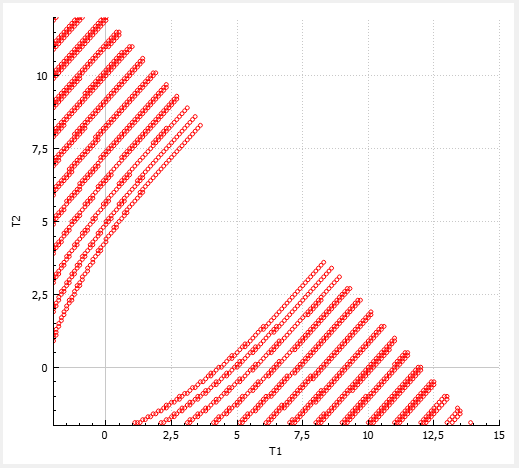


Рисунок 11 – 2D – график

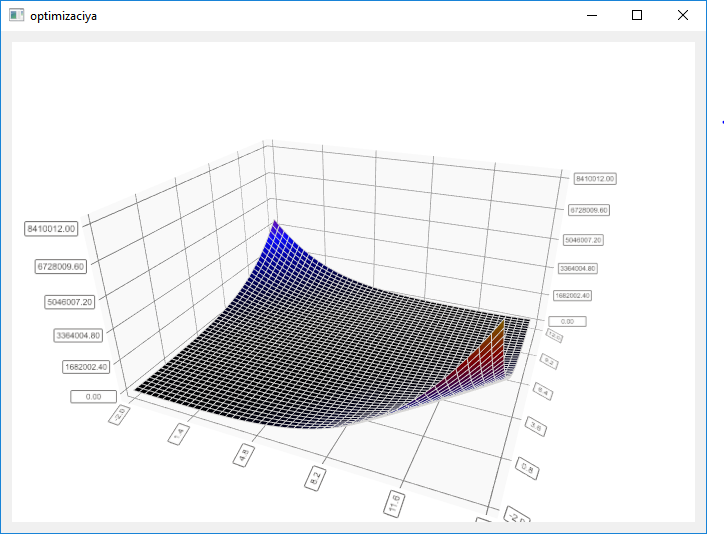


Рисунок 12 – 3D - график

# **Заключение**

Целью данного курсового проекта является создание прикладного программного комплекса под операционную систему семейства Windows для решения оптимизационных задач. В ходе разработки данного приложения необходимо необходимо было закрепить знания о процессе математического моделирования оптимизационных задач, о методах и особенностях их решения, полученные на лабораторных работах по дисциплине «Методы оптимизации в автоматизированных системах».

Во время разработки программного продукта были получены знания и навыки работы с 2D и 3D графикой, создания интерфейсной и программной части продукта, а также навыки по использованию математического аппарата для формализации и решения задач оптимизации.

**Список используемой источников**

1. Карпенко, А. П. Современные алгоритмы поисковой оптимизации. Алгоритмы, вдохновленные природой: учеб. пособие / А. П. Карпенко. – М.: Изд-во МГТУ им. Н. Э. Баумана, 2014. – 447 с.
2. Норенков, И. П. Автоматизированные информационные системы: учеб. пособие для вузов / И. П. Норенков. – М.: Изд-во МГТУ им. Н. Э. Баумана, 2011. – 342 с.
3. Смирнов И. А. Методы оптимизации. Базовый курс: учеб. пособие / И. А. Смирнов. – СПб.: СПбГТИ(ТУ), 2010. – 101 с.
4. ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.
5. Портал: Киберфорум [Электронный ресурс]: Электронные данные – Режим доступа: http://cyberforum.ru, свободный.
6. Портал: QtDoc[Электронный ресурс]: Электронные данные – Режим доступа: https://www.qt.io/, свободный.

# **Приложение А. Листинг программы**

main.cpp

#include "mainwindow.h"

#include <QApplication>

#include <QtDataVisualization/Q3DSurface>

#include <QtDataVisualization/QSurfaceDataProxy>

#include <QtDataVisualization/QHeightMapSurfaceDataProxy>

#include <QtDataVisualization/QSurface3DSeries>

#include <QtWidgets/QSlider>

#include <QtWidgets/QWidget>

#include <QtWidgets/QHBoxLayout>

#include <QtWidgets/QVBoxLayout>

#include <QtWidgets/QPushButton>

#include <QtWidgets/QRadioButton>

#include <QtWidgets/QSlider>

#include <QtWidgets/QGroupBox>

#include <QtWidgets/QComboBox>

#include <QtWidgets/QLabel>

#include <QtWidgets/QMessageBox>

#include <QtGui/QPainter>

#include <QtGui/QScreen>

#include "avtorization.h"

using namespace QtDataVisualization;

int **main**(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

if(!w.returnfla())

w.close();

return a.exec();

}

mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QtMath>

#include <vector>

#include<QStandardItemModel>

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit **MainWindow**(QWidget \*parent = 0);

~***MainWindow***();

bool fla;

bool **returnfla**();

private slots:

void **on\_pushButton\_clicked**();

void **on\_pushButton\_2\_clicked**();

void **on\_pushButton\_3\_clicked**();

void **on\_action\_3\_triggered**();

void **on\_action\_2\_triggered**();

void **on\_action\_4\_triggered**();

void **on\_comboBox\_2\_currentIndexChanged**(const QString &arg1);

private:

Ui::MainWindow \*ui;

double alfa=1,beta=1,gama=1;

double G=1;

double H=9;

double N=10;

double X1,X2;

double S,P,C=100;

double minX1 = -2, maxX1 = 15, minX2 = -2, maxX2 = 12, SUMX1X2 = 12;

double **ObjectiveFunction**(double,double);

double step = 1;

double FunctionMAX=0;

double X1P,X2P;

void **FindX1X2**();

std:: vector<std::vector<double>> Table;

void **SetValue**();

double kriteriyTochnost=10;

int **Okruglenie**(double);

QStandardItemModel \*tablemodel;

std::vector<double> \_vectorT1, \_vectorT2, \_vectorP;

void **GetDataFor2DGrafik**();

void **Bild2DGrafik**();

void **GetData**();

};

#endif // MAINWINDOW\_H

mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "surfacegraph.h"

#include "avtorization.h"

#include "dialog.h"

#include <QLocale>

#include <QDoubleValidator>

#include <QtWidgets/QHBoxLayout>

#include <QtWidgets/QVBoxLayout>

QLocale c\_locale("C");

MainWindow::**MainWindow**(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

Avtorization avtow;

avtow.setModal(true);

avtow.*exec*();

fla = avtow.flag;

if(!fla)

this->close();

Table.resize(3);

ui->lineEdit->setText(QString::number(minX1));

ui->lineEdit\_2->setText(QString::number(maxX1));

ui->lineEdit\_3->setText(QString::number(minX2));

ui->lineEdit\_4->setText(QString::number(maxX2));

ui->lineEdit\_6->setText(QString::number(SUMX1X2));

ui->lineEdit\_7->setText(QString::number(step));

ui->comboBox->addItem("P = α\*G((T1-T2)^2 + β\*(1/H)\*(T1+T2-γ\*N)^2)^2 ");

ui->comboBox\_2->addItem("Метод полного перебора");

ui->comboBox\_2->addItem("Метод золотого сечения");

ui->comboBox\_2->addItem("Метод половинного деления");

auto validatr = new QDoubleValidator(this);

validatr->setLocale(c\_locale);

ui->lineEdit->setValidator(validatr);

ui->lineEdit\_2->setValidator(validatr);

ui->lineEdit\_3->setValidator(validatr);

ui->lineEdit\_4->setValidator(validatr);

ui->lineEdit\_6->setValidator(validatr);

ui->lineEdit\_7->setValidator(validatr);

}

bool MainWindow::**returnfla**()

{

return fla;

}

void MainWindow::**GetData**()

{

auto replace = [] (QLineEdit\* e)

{

e->setText(e->text().replace(',','.'));

};

replace(ui->lineEdit);

replace(ui->lineEdit\_2);

replace(ui->lineEdit\_3);

replace(ui->lineEdit\_4);

replace(ui->lineEdit\_6);

replace(ui->lineEdit\_7);

minX1 = ui->lineEdit->text().toDouble();

maxX1 = ui->lineEdit\_2->text().toDouble();

minX2 = ui->lineEdit\_3->text().toDouble();

maxX2 = ui->lineEdit\_4->text().toDouble();

SUMX1X2 = ui->lineEdit\_6->text().toDouble();

step = ui->lineEdit\_7->text().toDouble();

}

double MainWindow::**ObjectiveFunction**(double X1, double X2)

{

P=alfa\*G\*qPow(qPow((X1-X2),2)+beta\*(1/H)\*(qPow((X1+X2-gama\*N),2)),2);

S=P\*C;

S=(qFloor(S/kriteriyTochnost))\*kriteriyTochnost;

return S;

}

void MainWindow::**FindX1X2**()

{

FunctionMAX=ObjectiveFunction(minX1,minX2);

for(X2=minX2; X2<=maxX2;X2+=step)

for(X1=minX1;X1<=maxX1;X1+=step)

if(X1+X2<=SUMX1X2)

if(ObjectiveFunction(X1,X2)>FunctionMAX)

{

X1P=X1;

X2P=X2;

FunctionMAX=ObjectiveFunction(X1,X2);

Table[0].insert(Table[0].end(),X1P);

Table[1].insert(Table[1].end(),X2P);

Table[2].insert(Table[2].end(),FunctionMAX);

}

}

MainWindow::~***MainWindow***()

{

delete ui;

}

int MainWindow::**Okruglenie**(double number)

{

int razryad=0;

int peremennaya;

number=number\*10;

peremennaya=qFloor(number);

while (peremennaya%10!=0) {

razryad++;

number=number\*10;

peremennaya=qFloor(number);

}

return razryad;

}

void MainWindow::**SetValue**()

{

}

void MainWindow::**on\_pushButton\_clicked**()

{

GetData();

for(int col=0;col<3;col++)

Table[col].clear();

FindX1X2();

ui->statusBar->showMessage("FMAX="+QString::number(FunctionMAX,'f',3)+" X1="+QString::number(X1P)+" X2="+QString::number(X2P));

tablemodel = new QStandardItemModel(Table[0].size(),Table.size(),this);

ui->tableView->*setModel*(tablemodel);

tablemodel->*setHeaderData*(0,Qt::Horizontal,"X1");

tablemodel->*setHeaderData*(1,Qt::Horizontal,"X2");

tablemodel->*setHeaderData*(2,Qt::Horizontal,"Function");

QModelIndex tablemodelindex;

int \*Okrugle = new int[Table.size()];

for(int col=0;col<tablemodel->*columnCount*();col++)

{

Okrugle[col]=Okruglenie(Table[col][1]);

}

for(int col=0;col<tablemodel->*columnCount*();col++)

{

for(int row=0;row<tablemodel->*rowCount*();row++)

{

tablemodelindex=tablemodel->*index*(row,col);

tablemodel->*setData*(tablemodelindex,QString::number(Table[col][row],'f',Okrugle[col]));

}

}

}

void MainWindow::**on\_pushButton\_2\_clicked**()

{

on\_pushButton\_clicked();

GetDataFor2DGrafik();

Bild2DGrafik();

}

void MainWindow::**GetDataFor2DGrafik**()

{

\_vectorT1.clear();

\_vectorT2.clear();

std::vector <double> \_LineLevel(Table[2]);

double \_step = 0.1;

double error = 0.05;

for(double \_meanLineLevel:\_LineLevel)

{

for(double \_indexT1 = minX1;\_indexT1<=maxX1;\_indexT1+=\_step)

{

for(double \_indexT2 = minX2; \_indexT2<=maxX2;\_indexT2+=\_step)

{

if(\_indexT1+\_indexT2<=SUMX1X2)

{

int \_P=alfa\*G\*qPow(qPow((\_indexT1-\_indexT2),2)+beta\*(1/H)\*(qPow((\_indexT1+\_indexT2-gama\*N),2)),2);

int \_S=\_P\*C;

if(\_S<=\_meanLineLevel\*(1+error)&&\_S>=\_meanLineLevel\*(1-error))

{

\_vectorT1.push\_back(\_indexT1);

\_vectorT2.push\_back(\_indexT2);

}

}

}

}

}

}

void MainWindow::**Bild2DGrafik**()

{

QVector <double> \_T1 = QVector<double>::fromStdVector(\_vectorT1),\_T2 = QVector<double>::fromStdVector(\_vectorT2);

ui->widget->clearGraphs();

ui->widget->addGraph();

ui->widget->graph(0)->setData(\_T1,\_T2);

ui->widget->graph(0)->setPen(QColor(255,0,0,255));

ui->widget->graph(0)->setLineStyle(QCPGraph::lsNone);

ui->widget->graph(0)->setScatterStyle(QCPScatterStyle(QCPScatterStyle::ssCircle, 4));

ui->widget->xAxis->setLabel("T1");

ui->widget->yAxis->setLabel("T2");

ui->widget->xAxis->setRange(minX1,maxX1);

ui->widget->yAxis->setRange(minX2,maxX2);

ui->widget->replot();

}

void MainWindow::**on\_pushButton\_3\_clicked**()

{

GetData();

Q3DSurface \*graph = new Q3DSurface();

QWidget \*container = QWidget::createWindowContainer(graph);

QSize screenSize = graph->screen()->size();

container->setMinimumSize(QSize(screenSize.width() / 2, screenSize.height() / 1.6));

container->setMaximumSize(screenSize);

container->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Expanding);

container->setFocusPolicy(Qt::StrongFocus);

QWidget \*widget = new QWidget;

QHBoxLayout \*hLayout = new QHBoxLayout(widget);

QVBoxLayout \*vLayout = new QVBoxLayout();

hLayout->addWidget(container, 1);

hLayout->addLayout(vLayout);

vLayout->setAlignment(Qt::AlignTop);

widget->show();

SurfaceGraph \*modifier = new SurfaceGraph(graph);

modifier->T1Max = maxX1;

modifier->T1Min = minX1;

modifier->T2Max = maxX2;

modifier->T2Min = minX2;

modifier->SUMT1T2 = SUMX1X2;

modifier->Special();

}

void MainWindow::**on\_action\_3\_triggered**()

{

QApplication::quit();

}

void MainWindow::**on\_action\_2\_triggered**()

{

Avtorization avtow;

avtow.setModal(true);

avtow.*exec*();

}

void MainWindow::**on\_action\_4\_triggered**()

{

Dialog windowTask;

windowTask.setModal(true);

windowTask.*exec*();

}

void MainWindow::**on\_comboBox\_2\_currentIndexChanged**(const QString &arg1)

{

ui->comboBox\_2->setCurrentText("Метод полного перебора");

}

surfacegraph.h

#ifndef SURFACEGRAPH\_H

#define SURFACEGRAPH\_H

#include <QtDataVisualization/Q3DSurface>

#include <QtDataVisualization/QSurfaceDataProxy>

#include <QtDataVisualization/QHeightMapSurfaceDataProxy>

#include <QtDataVisualization/QSurface3DSeries>

#include <QtWidgets/QSlider>

using namespace QtDataVisualization;

class SurfaceGraph : public QObject

{

Q\_OBJECT

public:

explicit **SurfaceGraph**(Q3DSurface \*surface);

~***SurfaceGraph***();

float T1Min = -2;

float T1Max = 15;

float T2Min = -2;

float T2Max = 12;

float SUMT1T2 = 12;

int CountT1 = 50;

int CountT2 = 50;

void **Special**();

void **toggleModeItem**() { m\_graph->setSelectionMode(QAbstract3DGraph::SelectionItem); }

private:

Q3DSurface \*m\_graph;

QSurfaceDataProxy \*m\_sqrtSinProxy;

QSurface3DSeries \*m\_sqrtSinSeries;

float m\_rangeMinX;

float m\_rangeMinZ;

float m\_stepX;

float m\_stepZ;

int m\_heightMapWidth;

int m\_heightMapHeight;

double Y;

void **setAxisXRange**(float min, float max);

void **setAxisZRange**(float min, float max);

void **fillSqrtSinProxy**();

};

#endif // SURFACEGRAPH\_H

surfacegraph.cpp

#include "surfacegraph.h"

#include <QtDataVisualization/QValue3DAxis>

#include <QtDataVisualization/Q3DTheme>

#include <QtGui/QImage>

#include <QtCore/qmath.h>

using namespace QtDataVisualization;

SurfaceGraph::**SurfaceGraph**(Q3DSurface \*surface)

: m\_graph(surface)

{

}

void SurfaceGraph::**Special**()

{

m\_graph->setAxisX(new QValue3DAxis);

m\_graph->setAxisY(new QValue3DAxis);

m\_graph->setAxisZ(new QValue3DAxis);

//! [0]

m\_sqrtSinProxy = new QSurfaceDataProxy();

m\_sqrtSinSeries = new QSurface3DSeries(m\_sqrtSinProxy);

//! [0]

fillSqrtSinProxy();

m\_sqrtSinSeries->setDrawMode(QSurface3DSeries::DrawSurfaceAndWireframe);

m\_sqrtSinSeries->setFlatShadingEnabled(true);

m\_graph->axisX()->setLabelFormat("%.1f");

m\_graph->axisZ()->setLabelFormat("%.1f");

m\_graph->axisX()->setRange(T1Min, T1Max);

m\_graph->axisY()->setRange(0.0f, Y);

m\_graph->axisZ()->setRange(T2Min, T2Max);

m\_graph->axisX()->setLabelAutoRotation(30);

m\_graph->axisY()->setLabelAutoRotation(90);

m\_graph->axisZ()->setLabelAutoRotation(30);

m\_graph->addSeries(m\_sqrtSinSeries);

QLinearGradient gr;

gr.setColorAt(0.0, Qt::black);

gr.setColorAt(0.33, Qt::blue);

gr.setColorAt(0.67, Qt::red);

gr.setColorAt(1.0, Qt::yellow);

m\_graph->seriesList().at(0)->setBaseGradient(gr);

m\_graph->seriesList().at(0)->setColorStyle(Q3DTheme::ColorStyleRangeGradient);

toggleModeItem();

}

SurfaceGraph::~***SurfaceGraph***()

{

delete m\_graph;

}

void SurfaceGraph::**fillSqrtSinProxy**()

{

float alfa=1,beta=1,gama=1;

float G=1;

float H=9;

float N=10;

float X1,X2;

float S,P,C=100;

float \_T1Max = T1Max;

float \_T2Max = T2Max;

float \_T1Min = T1Min;

float \_T2Min = T2Min;

int \_T1count = CountT1;

int \_T2count = CountT2;

float \_SUMT1T2 = SUMT1T2;

float stepT1 = (\_T1Max-\_T1Min)/float(\_T1count-1);

float stepT2 = (\_T2Max-\_T2Min)/float(\_T2count-1);

float \_z = \_T2Min;

float \_x = \_T1Min;

float YMax =0;

QSurfaceDataArray \*dataArray = new QSurfaceDataArray;

dataArray->reserve(\_T2count);

for (int i = 0 ; i < \_T2count ; i++) {

QSurfaceDataRow \*newRow = new QSurfaceDataRow(\_T1count);

int index = 0;

for (int j = 0; j < \_T1count; j++) {

P=alfa\*G\*qPow(qPow((\_x-\_z),2)+beta\*(1/H)\*(qPow((\_x+\_z-gama\*N),2)),2);

float \_y =P\*C;

(\*newRow)[index++].setPosition(QVector3D(\_x, \_y, \_z));

\_x += stepT1;

if(\_y>YMax)

YMax=\_y;

}

\*dataArray << newRow;

\_z += stepT2;

\_x = \_T1Min;

}

Y=YMax;

m\_sqrtSinProxy->resetArray(dataArray);

}

avtorization.h

#ifndef AVTORIZATION\_H

#define AVTORIZATION\_H

#include <QDialog>

namespace Ui {

class Avtorization;

}

class Avtorization : public QDialog

{

Q\_OBJECT

public:

explicit **Avtorization**(QWidget \*parent = 0);

~***Avtorization***();

bool **returnFlag**();

bool flag = false;

private slots:

void **on\_pushButton\_clicked**();

void **on\_pushButton\_2\_clicked**();

private:

Ui::Avtorization \*ui;

};

#endif // AVTORIZATION\_H

avtorization.cpp

#include "avtorization.h"

#include "ui\_avtorization.h"

#include <QMessageBox>

#include "mainwindow.h"

Avtorization::**Avtorization**(QWidget \*parent) :

QDialog(parent),

ui(new Ui::Avtorization)

{

ui->setupUi(this);

}

Avtorization::~***Avtorization***()

{

delete ui;

}

void Avtorization::**on\_pushButton\_clicked**()

{

QString pass = ui->lineEdit\_2->text();

QString login = ui->lineEdit->text();

if(login == "1234"&& pass =="1234")

{

flag = true;

this->close();

}

else

{QMessageBox::information(0,"Ошибка","Введены неправильные данные!!!","ОК");

}

}

bool Avtorization::**returnFlag**()

{

return flag;

}

void Avtorization::**on\_pushButton\_2\_clicked**()

{

QApplication::quit();

}

dialog.h

#ifndef DIALOG\_H

#define DIALOG\_H

#include <QDialog>

namespace Ui {

class Dialog;

}

class Dialog : public QDialog

{

Q\_OBJECT

public:

explicit **Dialog**(QWidget \*parent = 0);

~***Dialog***();

void **Exit**();

private slots:

void **on\_pushButton\_clicked**();

void **on\_pushButton\_2\_clicked**();

void **on\_pushButton\_3\_clicked**();

void **on\_pushButton\_4\_clicked**();

private:

Ui::Dialog \*ui;

};

#endif // DIALOG\_H

dialog.cpp

#include "dialog.h"

#include "ui\_dialog.h"

Dialog::**Dialog**(QWidget \*parent) :

QDialog(parent),

ui(new Ui::Dialog)

{

ui->setupUi(this);

}

Dialog::~***Dialog***()

{

delete ui;

}

void Dialog::**Exit**()

{

this->close();

}

void Dialog::**on\_pushButton\_clicked**()

{

Exit();

}

void Dialog::**on\_pushButton\_2\_clicked**()

{

Exit();

}

void Dialog::**on\_pushButton\_3\_clicked**()

{

Exit();

}

void Dialog::**on\_pushButton\_4\_clicked**()

{

Exit();

}