Министерство образования и науки Кыргызский Республики

**Кыргызский Государственный Технический Университет**

**им. И. Раззакова**

**Факультет информационных технологий**

**ПРОГРАММНАЯ ИНЖЕНЕРИЯ**

**Бакалаврский курс**

**Основы разработки и анализа требований к ПО**

**Отчет**

**по лабораторной работе №2**

**«Newton Method»**

Выполнил: Байгазиев Санжар

Группа ПИ-2-15

4 курс

Преподаватель: кандидат технических наук, профессор

Тен Иосиф Григорьевич

Бишкек 2018

Содержание

[Глава 1: Описание проблемы 3](file:///D:\lessons\4-курс\РАТПО\Лабы\NewtonMethod\NewtonMethod2.docx#_Toc525767936)

[Глава 2: Описание Newton Method 3](file:///D:\lessons\4-курс\РАТПО\Лабы\NewtonMethod\NewtonMethod2.docx#_Toc525767937)

[Глава 3: Спецификация требований к программному обеспечению 4](file:///D:\lessons\4-курс\РАТПО\Лабы\NewtonMethod\NewtonMethod2.docx#_Toc525767938)

[3.1. Наименование программной разработки 4](file:///D:\lessons\4-курс\РАТПО\Лабы\NewtonMethod\NewtonMethod2.docx#_Toc525767939)

[3.2. Описание бизнес-процесса и анализ расхождения 4](file:///D:\lessons\4-курс\РАТПО\Лабы\NewtonMethod\NewtonMethod2.docx#_Toc525767940)

[3.3. Бизнес-цель разработки программного обеспечения 4](file:///D:\lessons\4-курс\РАТПО\Лабы\NewtonMethod\NewtonMethod2.docx#_Toc525767941)

[3.4. Пользовательская история разработки программного обеспечения 4](file:///D:\lessons\4-курс\РАТПО\Лабы\NewtonMethod\NewtonMethod2.docx#_Toc525767942)

[3.5. Функциональные требования к разработке программного обеспечения 5](file:///D:\lessons\4-курс\РАТПО\Лабы\NewtonMethod\NewtonMethod2.docx#_Toc525767943)

[3.6. Нефункциональные требования на разработку программного обеспечения 6](file:///D:\lessons\4-курс\РАТПО\Лабы\NewtonMethod\NewtonMethod2.docx#_Toc525767944)

[3.7. Ограничения на разработку программного обеспечения 6](file:///D:\lessons\4-курс\РАТПО\Лабы\NewtonMethod\NewtonMethod2.docx#_Toc525767945)

[Глава 4: Стадии проектирования системы для поиска корня нелинейного уравнения f(x)=0, реализующей Newton Method: 8](file:///D:\lessons\4-курс\РАТПО\Лабы\NewtonMethod\NewtonMethod2.docx#_Toc525767946)

[Глава 5: Дизайн интерфейсной формы системы, реализующей Newton method: 11](file:///D:\lessons\4-курс\РАТПО\Лабы\NewtonMethod\NewtonMethod2.docx#_Toc525767947)

[Глава 6: Стадии конструирования системы для поиска корня нелинейного уравнения f(x)=0, реализующей Newton Method: 12](file:///D:\lessons\4-курс\РАТПО\Лабы\NewtonMethod\NewtonMethod2.docx#_Toc525767948)

[Stage №.1: Код программы на C#, ассоцированный с интерфейсной формой “-Form1.cs” 12](file:///D:\lessons\4-курс\РАТПО\Лабы\NewtonMethod\NewtonMethod2.docx#_Toc525767949)

[Stage No.2: Код программы на C#, ассоцированный с классом “NewtonMethod.cs” , который реализует логику Newton method по поиску корня нелинейного уравнения и составляет отдельный модуль в виде Public Class “ NewtonMethod ”: 16](file:///D:\lessons\4-курс\РАТПО\Лабы\NewtonMethod\NewtonMethod2.docx#_Toc525767950)

[Stage No.3: Код программы на C#, ассоцированный с ***программным модулем*** “Newton Method.cs” , который реализует логику Newton method по поиску корня нелинейного уравнения и показывает код по управлению Progress Bar: 18](file:///D:\lessons\4-курс\РАТПО\Лабы\NewtonMethod\NewtonMethod2.docx#_Toc525767951)

[Stage No.4: Подключение парсера 19](file:///D:\lessons\4-курс\РАТПО\Лабы\NewtonMethod\NewtonMethod2.docx#_Toc525767952)

[Результат работы программы, реализующей Bisection method: 21](file:///D:\lessons\4-курс\РАТПО\Лабы\NewtonMethod\NewtonMethod2.docx#_Toc525767953)

# Глава 1: Описание проблемы

Найти корень произвольного нелинейного уравнения – ноль нелинейной функции f(x) с заданной допустимой погрешностью Tolerance не более величины 1E–28 методом Ньютона (Newton Method). Нелинейная функция f(x) имеет произвольный аналитический вид, составленный из математических функций (полиномов различных степеней, тригонометрических – sin(x), cos(x), exp(x), ln(x), log(x) и. т. д.), которая имеет математический смысл, и для которой существует хотя бы одно решение задачи.

# Глава 2: Описание Newton Method

Для решения нелинейного уравнения *f(x)=0* по методу Ньютона используется итерационный процесс:

|  |  |
| --- | --- |
| *x(k+1) = x(k) - f(x(k))/f '(x(k)) , k = 0, 1, 2, ...* |  |

где *x(0)* - некоторое начальное приближение к корню.

При этом предполагается, что *f '(x)*≠ *0*

Newton method. Проиллюстрируем метод Ньютона, поиск корня нелинейного уравнения.

Функция f(x) = x2-4sin(x) = 0

Производная функции f’(x)=2x-4cos(x)

Итерационная схема

Принимая X0 = 3 в качестве начальной точки, мы получим последовательность итераций, показанных в следующем, где h = -f(x)/f’(x) означает изменение х на каждой итерации. Итерация завершается, когда |h| станет меньше по отношению к | x |.

|  |  |  |  |
| --- | --- | --- | --- |
| a | f(x) | f'(x) | h |
| 3,000000 | 8,435520 | 9,959970 | -0,846942 |
| 2,153058 | 1,294773 | 6,505772 | -0,199019 |
| 1,954039 | 0,108439 | 5,403795 | -0,020067 |
| 1,933972 | 0,001152 | 5,288920 | -0,000218 |
| 1,933754 | 0,000000 | 5,287670 | 0,000000 |

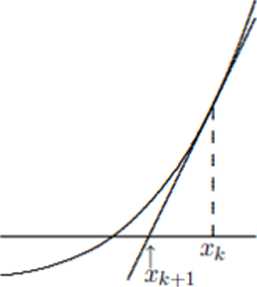


Рисунок 1: Метод Ньютона для решения нелинейного уравнения.

[GoTo Contents](file:///D:\\lessons\\4-курс\\РАТПО\\Лабы\\Bisection%20method_ver2\\Отчет.docx" \l "Contents)

# Глава 3: Спецификация требований к программному обеспечению

## 3.1. Наименование программной разработки

Система поиска корней нелинейного уравнения методом Ньютона

## 3.2. Описание бизнес-процесса и анализ расхождения

При решении различных бизнес-задач отделу исследований (users, clients, customers) требуется находить корень произвольного нелинейного уравнения с высокой точностью. Для решения таких задач в этом отделе имеется 5 ставок специалистов. Два из них занимаются формализацией задачи и определением/нахождением аналитического вида нелинейной функции, описывающей, как можно точнее, модель поведения реальной системы, а остальные занимаются решением задачи – поиском корня нелинейного уравнения для найденной нелинейной функции. При больших трудозатратах (заняты три специалиста высокой квалификации), не всегда обеспечивается приемлемая и единообразная/унифицированная точность решения задачи. Бизнесу желательно было бы гарантированно обеспечить любую допустимую погрешность решения задачи при любых видах нелинейной функции f(x) с минимальными трудозатратами.

Таким образом в отделе для достижения цели проводится следующие действия

1. формализуется задачи
2. определяется аналитический вид нелинейной функции
3. находят корня нелинейной функции

Для решения таких задач в отделе заняты три специалиста высокой квалификации и не всегда обеспечивается приемлемая точность решения задачи. После автоматизации, деятельность отдела существенно изменится, так как нахождением корня не линейного уравнения займется программа. Программа обеспечит любую допустимую погрешность до 1e-28 для решения задачи при любых видах нелинейной функции за короткое время.

## 3.3. Бизнес-цель разработки программного обеспечения

Минимизировать затраты времени на поиск корня нелинейного уравнения ***до одной минуты*** при максимально высокой точности решения – допустимая погрешность решения должна быть ***не менее 1e-28 (т.е., 0.000000000000000000000000001)***, что позволит сократить ***две ставки*** специалиста в отделе исследований, поскольку теперь с такой задачей может справиться уже один специалист с помощью разрабатываемой системы.

## 3.4. Пользовательская история разработки программного обеспечения

***Пользователь:***

* должен вводить аналитическое выражение для требуемой нелинейной функции ***f(x)***;
* начальную точку **x**;
* значение допустимой погрешности ***Tolerance*** решения задачи;
* допустимое максимальное количество ***k\_max*** итераций.
* допустимое максимальное количество ***max\_time*** времени

***Программа:***

* должна провести parsing – синтаксический анализ выражения введенной нелинейной функции и преобразовать аналитическую запись нелинейной функции в виде символов в вычисляемую процедуру в виде подпрограммы-функции;
* должна ввести символьную значению начальной точки **x** и преобразовать в числовое
* должна найти аналитическое выражение производной от fx
* должна провести parsing – синтаксический анализ выражения производной функции и преобразовать аналитическую запись нелинейной функции в виде символов в вычисляемую процедуру в виде подпрограммы-функции;
* должна найти решение задачи – корень нелинейного уравнения ***f(x)=0*** или ноль нелинейной функции ***f(x)*** – по алгоритму Newton Method;
* должна выводить аналитическое выражение производной функции
* должна выводить найденное значение ***X\**** корня заданного нелинейного уравнения;
* должна выводить значение заданной нелинейной функции ***f(x\*)*** в этой точке;
* должна выводить значение производной от функции ***f’(x\*)*** в этой точке;
* должна выводить общее затраченное время (в миллисекундах) на поиск решения ***Elapsed Time*** для оценки производительности программы;
* должна выводить затраченное на поиск решения с заданной погрешностью количество итераций ***Amount Of Iterations***;
* должна выводить достигнутое значение абсолютной ошибки решения ***Abs(b–a)***.

## 3.5. Функциональные требования к разработке программного обеспечения

* Система должна использовать ***Newton Method для поиска корней*** нелинейного уравнения;
  + Система должна искать решение задачи по следующему алгоритму (Newton algorithm):

Do{

Xn-1 = Xn

Xn+1 = Xn – (f(Xn)/f’{Xn)

K=k+1}

While(***Abs(Xn – Xn-1)***) Tol;

* + Система должна обеспечивать ввод любых начальных числовых значений начальной точки ***[X]***;
  + Система должна обеспечить возможность продолжения поиска решения задачи с последнем достигнутом на шаге ***k\_max*** значении точки ***[Xk(k\_max)]***, приняв ее за начальное значение начальной точки {т.е., система должна заново выполнить не менее ***k\_max*** итераций}, если за данное количество итераций ***k\_max*** не достигнуто решение с требуемой погрешностью ***Tolerance***;
* Система должна обеспечивать возможность ввода исходных данных имеющих следующие особенности:
  + Система должна обеспечивать возможность ввода ***аналитического выражения*** для любой нелинейной функции;
  + Система должна проводить ***parsing***\*\* – синтаксический анализ аналитического выражения нелинейной функции f(x) – и автоматически конструировать подпрограмму для вычисления значения этой функции для любого значения аргумента ***x*** этой функции;
  + Система должна проводить ***parsing***\*\* – синтаксический анализ аналитического выражения производной функции f(x) – и автоматически конструировать подпрограмму для вычисления значения этой функции для любого значения аргумента ***x*** этой функции;
  + Система должна обеспечить ввод любого целого числового значения в качестве максимального допустимого количества итераций ***k\_max***;
  + Система должна обеспечивать возможность ввода любых числовых значений допустимой погрешности ***Tolerance*** как в формате с фиксированной точностью “0.00000000000000000000000001”, так и в экспоненциальном формате “1e-28”;
  + Система должна быть способна проверять валидность введенного аналитического выражения в качестве функции f(x) и сообщать пользователю об ошибке в аналитическом выражении функции;
  + Система должна быть способна проверять валидность формата введенных исходных числовых данных и сообщать пользователю об ошибке в формате исходных данных;
* Система должна обеспечить вывод следующих результатов решения задачи:
  + Система должна обеспечить вывод результата решения задачи в виде числа x\* – корня нелинейного уравнения или нуля нелинейной функции;
  + Система должна обеспечить вывод значения функции в этой точке f(x\*);
  + Система должна обеспечить вывод значения производной f’(x);
  + Система должна обеспечить вывод количества итераций, которые потребовались системе, чтобы найти решение задачи с требуемой погрешностью в качестве критерия производительности системы;
  + Система должна обеспечить вывод достигнутого значения погрешности решения задачи в качестве оценки точности этого метода (Newton Method);
  + Система должна обеспечить невозможность какого-либо ***исправления вручную*** выходных данных системы;
* Система должна обеспечить ***оценку производительности программы*** при решении задачи заданным методом (***Newton Method***) и реализации программы на языке ***C# в среде Visual Studio 2017*** путем измерения и вывода времени, затрачиваемой на поиск решения с заданной погрешностью;

## 3.6. Нефункциональные требования на разработку программного обеспечения

* Программа в процессе поиска решения должна отображать индикатор состояния процесса выполнения операции ***Progress Bar***, чтобы пользователь понимал, что система не зависла, а находится в рабочем состоянии;
* В случае ошибки во введенном аналитическом выражении нелинейной функции f(x) система должна предоставлять достаточно подробную информацию об ошибке, которая будет облегчать поиск источника ошибки;
* Система должна обеспечить возможность очищения текстовых полей ***выходного*** интерфейса от результатов предыдущих вычислений при новом запуске вычислений. При этом данные в текстовых полях ***входного*** интерфейса должны сохраняться. Это позволит пользователю увидеть более наглядно, что получены новые результаты, соответствующие измененным входным данным;
* Сбой системы не должен происходить чаще чем один раз на тысячу запусков программы;
* Система должна обеспечить оценку качества решения задачи заданным методом (***Newton Method***) путем измерения и вывода количества итераций, потраченных на поиск решения с заданной погрешностью;

## 3.7. Ограничения на разработку программного обеспечения

* Нелинейная функция ***f(x)*** должна быть непрерывной в точке x0;
* Разработать ПО на основе IDE (integrated development environment – интегрированная среда разработки) – Visual Studio 2017;
* Разработать ПО на языке C#;
* Система должна решать задачи с допустимой погрешностью до 1e-28;
* Система должна решать задачи за время не более одной минуты;
* Временные затраты на разработку ПО не должны превышать одного месяца;
* Финансовые затраты на разработку ПО и соответствующую документацию (техническое задание на ПО, руководство пользователя и руководство программиста) не должны превышать $1000;
* Система должна быть разработана для эксплуатации на компьютерах типа пентиум с оперативной памятью не более 2 гигабайт;
* Разработка программы должна начаться не позже 10 сентября 2018 года, и система должна быть передана заказчику не позднее 1 октября 2018 года;
* Система должна позволять пользователю решать не менее 100 задач поиска корня нелинейного уравнения за один час для любой нелинейной функции, описываемой в виде аналитического выражения, корректной с точки зрения математики;
* Система должна понимать и обрабатывать любые аналитические выражения для нелинейной функции, которые включают следующие математические символы:

Cстандартных функций

"sqrt", "sin", "cos", "tan",

"atan", "acos", "asin", "acotan",

"exp", "ln", "log",

"sinh", "cosh", "tanh", "abs",

"ceil", "floor", "fac", "sfac", "round", "fpart"

* + Основных вычислительных операций: сложение (+), вычитание (-), умножение (\*), деление (/), остаток от деления (%) и возведение в степень (^);
  + круглых и квадратных скобок любой вложенности;
  + вектора переменных, на которые можно ссылаться, используя запись вида x[N] или x(N), где N - индекс переменной.

[GoTo Contents](file:///D:\lessons\4-курс\РАТПО\Лабы\Bisection%20method_ver2\Отчет.docx#Contents)