|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Робототехники и комплексной автоматизации

КАФЕДРА Системы автоматизированного проектирования (РК-6)

**ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

Студент Ошкина Елена Владимировна

Группа РК6-32М

Тип задания лабораторная работа

Тема лабораторной работы Графовые модели

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_Ошкина Е.В. \_\_\_\_\_**

*подпись, дата фамилия, и.о.*

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_Соколов А. П. \_\_\_\_\_**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Москва, 2018 г.*

# Задание на лабораторную работу

Разработать шаблон файла исходных данных и графовую модель программной реализации заданного вычислительного метода.

Требуется:

1. разработать графовую модель, позволяющую реализовать масштабируемую программную реализацию заданного вычислительного метода, включая определение наименований и описаний: состояний данных, функций – предикатов и функций обработчиков.
2. Обосновать, что построенная графовая модель действительно будет реализовывать масштабируемую программную реализацию заданного вычислительного метода.
3. Указать, какие функции – предикаты будут связаны друг с другом, что требует изменения стратегии распараллеливания в узле.
4. Разработать текстовый файл в формате aINI исходных данных для подачи на вход «решателю», который будет реализован на основе созданной графовой модели.
5. Представить графическое представление построенной графовой модели в отчете о лабораторной работе.

Оглавление

[Задание на лабораторную работу 2](#_Toc532458293)

[Вариант 7 4](#_Toc532458294)

[Аналитическое решение и получение общего вида решения 5](#_Toc532458295)

[Постановка обратной задачи общего вида 6](#_Toc532458296)

[Явное задание функции на основе известного общего вида решения 7](#_Toc532458297)

[Построение графовой модели программной реализации вычислительного метода решения обратной задачи 8](#_Toc532458298)

[Построение графовой модели метода решения задачи оптимизации методом Монте-Карло 8](#_Toc532458299)

[Алгоритм метода Монте-Карло 8](#_Toc532458300)

[Графовая модель метода решения задачи оптимизации 9](#_Toc532458301)

[Определение файла входных данных задачи в формате aINI 12](#_Toc532458302)

[Графовая модель численного метода решения прямой задачи, в качестве метода решения используется метод Рунге-Кутты 4 порядка точности. 13](#_Toc532458303)

[Алгоритм метода Рунге-Кутты 4 порядка 13](#_Toc532458304)

[Графовая модель численного метода решения 16](#_Toc532458305)

[Определение файла входных данных задачи в формате aINI 17](#_Toc532458306)

[Заключение 18](#_Toc532458307)

[Список использованных источников 19](#_Toc532458308)

# Вариант 7

Графовая модель метода решения обратной задачи идентификации параметров ОДУ

В рамках общей формулировки задачи, пусть задано ОДУ:



Где L[ – дифференциальный оператор, – некоторая заданная функция, – вектор внутренних параметров модели.

Формулировка **прямой** задачи:

;

+ + ;

где – неизвестная функция, тогда как g известных константы.

# Аналитическое решение и получение общего вида решения

;

Допустим:

Вынесем *:*

Так как правый множитель не может быть равен 0 для любого значения

Первый корень дает:

, где – произвольная константа

, где – произвольная константа

Общее решение:

Найдем частное решение методом неопределенных коэффициентов для:

Решение имеет форму:

Вычислим для неизвестной константы:

Подставим частное решение в дифференциальное уравнение:

Подставим в

Решение:

# Постановка обратной задачи общего вида

Формулировка обратной задачи в рамках обозначений соответствующей формулировки прямой задачи:

;

*где - известная функция, требуется определить.*

# Явное задание функции на основе известного общего вида решения

Функция задается явно на основе известного вида решения при условии, что параметры принадлежат указанным диапазонам и задаются случайным образом (далее фиксированы для задачи):

;

;

;

;

;

;

Для параметров были выбраны следующие значения: Явное задание функции:

# Построение графовой модели программной реализации вычислительного метода решения обратной задачи

# Построение графовой модели метода решения задачи оптимизации методом Монте-Карло

# Алгоритм метода Монте-Карло

В рамках лабораторной работы необходимо решить следующую задачу глобальной условной оптимизации:

где множество допустимых значений которые в свою очередь принадлежат указанным диапазонам:

;

Схема метода Монте-Карло

1. Задаем общее количество испытаний *N*  и полагаем счетчик числа итераций *r* = 1.
2. С помощью какого-либо программного генератора случайных чисел генерируем  компоненты вектора
3. Вычисляем Ф( и полагаем ,
4. Аналогично пункту 2 генерируем случайные точки для вектора . Вычисляем соответствующее значение критерия оптимальности Ф(
5. Выполняем следующие присваивания:
6. Если полагаем и переходим на пункт 4, иначе принимаем , в качестве приближенного решения задачи и заканчиваем вычисления.

Отметим, что в простейшем случае точки вектора генерируются равномерно распределенными в области . С целью сокращения вычислительных затрат и при наличии априорной информации о положении точки глобального минимума, целесообразно использовать законы распределения, в которых вероятность генерации точки в окрестности предполагаемо глобального минимума выше, чем вне этой окрестности [3].

# Графовая модель метода решения задачи оптимизации

Для описания графовой модели использовался язык aDOT. Результат описания представляет собой текстовый файл с расширением .gv или .dot. Полученное описание представлено в листинге 1.

Листинг 1. Описание графовой модели

|  |
| --- |
| digraph G{  INPUT\_READY -> GENERATED\_N\_COMPONENTS [label = "генерируем n компонент"];  GENERATED\_N\_COMPONENTS -> CALCULATED\_FITNESS\_FUNCTION [label = "вычисляем значение фитнес-функции"];  CALCULATED\_FITNESS\_FUNCTION ->CHANGED\_VALUES\_VECTOR\_G [label = "меняем текущее значение на вычисленное"];  CHANGED\_VALUES\_VECTOR\_G -> GENERATED\_POINT[label = "генерируем новую случайную точку"];  GENERATED\_POINT -> CALCULATED\_FITNESS\_FUNCTION [label = "вычисляем значение фитнес-функции"];  CALCULATED\_FITNESS\_FUNCTION ->DID\_ASSIGN[label = "выполняем присваивания"];  DID\_ASSIGN -> CHECKED\_VALUE\_R [label = "проверка значения шага и его изменение"];  CHECKED\_VALUE\_R -> CHANGED\_STEP\_VALUE [label = "генерируем новую случайную точку"];  CHANGED\_STEP\_VALUE->GENERATED\_POINT [label = "генерируем новую случайную точку"];  CHECKED\_VALUE\_R->TASK\_RESOLVED [label = "принимаем полученные значения в качестве оптимальных"];  } |

# Определение файла входных данных задачи в формате aINI

Листинг 2. Описание входных данных задачи

|  |
| --- |
| [Input data]  a=[@alpha@;1:20;1] // Параметр alpha  b=[@betta@;2:10;1] // Параметр betta  g=[@gamma@;3:15;1] // Параметр gamma  max\_iteration = 1000 //максимальное количество итераций  EPS = 0.001 //требуемая точность  Accuracy = 3 //количество цифр после запятой |

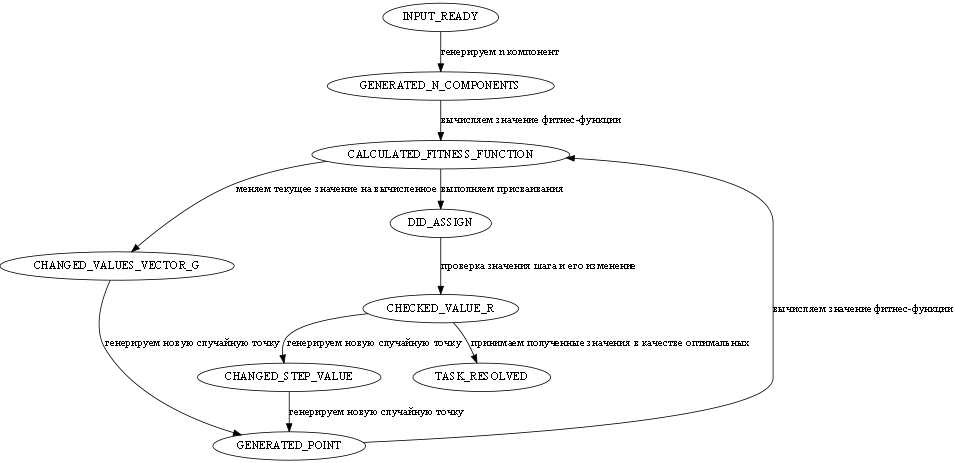


Рисунок 1. Графовая модель метода оптимизации Монте-Карло

# Графовая модель численного метода решения прямой задачи, в качестве метода решения используется метод Рунге-Кутты 4 порядка точности.

# Метод Рунге-Кутты 4 порядка

Классический метод Рунге-Кутта 4-го порядка описывается следующей системой пяти неравенств:

, где

Блок схема алгоритма представлена на рисунке 1.

Строго говоря, существует не один, а группа методов Рунге-Кутты, отличающихся друг от друга порядком, то есть количеством параметров В данном случае мы имеет метод 4-го порядка, который является одним из наиболее применяемых на практике, так как обеспечивает высокую точность и в то же время отличается сравнительной простотой [5].

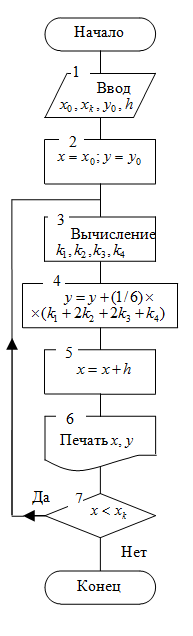


Рисунок 2. Блок-схема алгоритма Рунге-Кутты

# Листинг 3. Описание входных данных задачи

|  |
| --- |
| digraph G{  size="7,7";  INPUT\_READY -> DETERMINED\_COEFFICIENTS [label = "определяем коэффициенты"];  DETERMINED\_COEFFICIENTS -> CONSTRUCTED\_SEQUENCE\_VALUES [label="строим последовательность значений функции"];  CONSTRUCTED\_SEQUENCE\_VALUES -> COMPARED\_WITH\_ANALYTICAL\_SOLUTION [label="сравнение с аналитическим решением"];  COMPARED\_WITH\_ANALYTICAL\_SOLUTION ->CHECKED\_TARGET\_ACCURACY [label="проверка достигнутой точности"];  COMPARED\_WITH\_ANALYTICAL\_SOLUTION ->MAX\_NUMBER\_ITERATION\_REACHED [label="проверка достижения макс количества итераций"];  MAX\_NUMBER\_ITERATION\_REACHED -> ANALYZED\_ACCYRACY\_AND\_N [label="анализ полученных значений n и eps"];  ANALYZED\_ACCYRACY\_AND\_N ->SAVED\_RESULTS\_TO\_FILE [label="вывод полученноного вектора значений в файл"];  CHECKED\_TARGET\_ACCURACY ->ANALYZED\_ACCYRACY\_AND\_N [label="анализ полученных значений n и eps"];  ANALYZED\_ACCYRACY\_AND\_N -> REDUCED\_STEP [label="уменьшаем размер шага"];  REDUCED\_STEP -> DETERMINED\_COEFFICIENTS [label = "определяем коэффициенты"];  } |

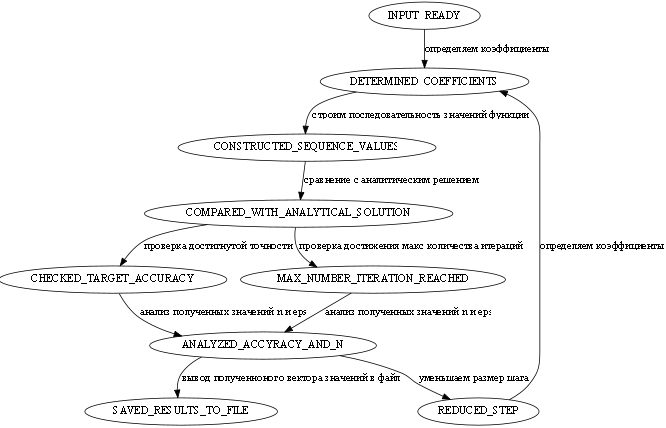


Рисунок 3. Графовая модель численного метода Рунге-Кутты

# Определение файла входных данных задачи в формате aINI

Листинг 4. Описание графовой модели

|  |
| --- |
| [Input data]  Method = runge\_k  a = 3  b = 4  g = 8  a\_min = 1  a\_max = 20  b\_min = 2  b\_max = 10  g\_min = 3  g\_max = 15  max\_iteration = 1000 //максимальное количество итераций  result = @vector\_results@  EPS = 0.001 //требуемая точность  Accuracy = 3 //количество цифр после запятой  resFileName = @runge\_k@.geo //Файл результатов |

# Заключение

Разработаны шаблоны файлов исходных данных и графовые модели программных реализаций методов Монте-Карло и Рунге-Кутты.

Списки входных параметров и их значений были заданы с помощью файла в формате aINI. Использование данного формата позволило полностью отделить процессы подготовки исходных данных от процессов их обработки. Рассмотренный в лабораторной работе программный инструментарий продемонстрировал возможность независимой разработки вычислительных программных модулей и подготовки входных данных.

# Список использованных источников

1. Соколов, А.П., Першин, А.Ю. Инструкция по выполнению лабора-

торной работы (общая). Москва: Соколов, А.П., Першин, А.Ю., 2018.

с. 4.

2. Соколов А.П., Першин А.Ю. Программный инструментарий для со-

здания подсистем ввода данных при разработке систем инженерного

анализа // Программная инженерия. 2017. Т. 8, No 12. С. 543–555

3. Соколов А.П. Распределенная вычислительная система GCD:

Формат данных Advanced INI (aINI).

2007-2017. URL:

https://archrk6.bmstu.ru/index.php/f/100757.

4. Соколов А.П. Программные технологии разработки систем ин-

женерного анализа / Лекция 7. Методика графоориентирован-

ной разработки ПО. Технология GBSE.

2016. 11. URL:<https://archrk6.bmstu.ru/index.php/f/2821>.

5. Исаков В. Б. Элементы численных методов: Учебное пособие для студентов обущающихся по специалиности Математика. –М. Академия, 2003.- 192 с.

# 