МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
Факультет э***лектротехнический***
Кафедра ***автоматики***
Направление подготовки ***27.03.04 «Управление в технических системах»***

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой
_____
_____Голощапов С.С._____
_____2021 г.

Исследование алгоритмов и моделирование расчета параметров движения объекта
(тема)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ – 27.03.04.2021. 203.03.165 ПЗ ВКР

Автор проекта
студент группы МиЭт-426

_____/__Карапетян А.А._____
подпись ФИО
_____2021 г.

Руководитель работы
_д.ф.-м.н., профессор_____
должность
_____/___Телегин А.И._____
подпись ФИО
_____2021 г.

Нормоконтроль

__ст.преподаватель_____
должность
_____/__Елисеев В.П._____
подпись ФИО
_____2021 г.

Миасс 2021 г.

АННОТАЦИЯ

Карапетян А.А. «Исследование алгоритмов и моделирование расчета параметров движения объекта». Миасс: ЮУрГУ, Автоматика, 2021 г. 86 стр., 8 ил., 1 табл., библиогр. список - 10 наим., 4 приложения.

- Цель ВКР

Исследование основных свойств параметров движения объекта в гравитационном поле Земли, изучение методов численного интегрирования дифференциальных уравнений, используемых в баллистике космических летательных аппаратов (КЛА), разработка методики сравнительной оценки применения методов Рунге-Кутты для решения задачи прогноза движения КЛА на участке свободного полета в гравитационном поле Земли без учета влияния атмосферы и проведение с ее использованием исследований для определения оптимального метода по критерию «заданная точность-минимальные вычислительные затраты»

- Результат ВКР

Для заданных начальных условий движения КЛА и заданных требований к точности решения задачи прогноза движения КЛА проведен сравнительный анализ алгоритмов, разработанных на основе выбранных методов. Определен оптимальный алгоритм, получена оценка его эффективности, на основании которой он может быть рекомендован для использования в вычислительных комплексах бортовых и наземных системах управления полетом КЛА.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 7

1 ПОСТРОЕНИЕ МАТЕМАТИЧЕСКОЙ МОДЕЛИ 9

1.1 Основные положения 9

1.2 Уравнения движения КЛА в векторной форме 10

1.3 Системы координат 11

1.4 Система дифференциальных уравнений движения летательного аппарата 12

2 ЧИСЛЕННЫЕ МЕТОДЫ РУНГЕ-КУТТЫ 15

2.1 Метод Рунге-Кутты второго порядка 15

2.2 Методы Рунге-Кутты высоких порядков 16

3 МЕТОДИКА СРАВНИТЕЛЬНОГО АНАЛИЗА АЛГОРИТМОВ 19

4 АЛГОРИТМ ИНТЕГРИРОВАНИЯ МЕТОДАМИ РУНГЕ-КУТТЫ РАЗНЫХ ПОРЯДКОВ 21

4.1 Метод Рунге-Кутты-Мерсона с шагом $h = 1$ 23

4.2 Метод Рунге-Кутты второго порядка. 26

4.3 Метод Рунге-Кутты третьего порядка 28

4.4 Метод Рунге-Кутты четвертого порядка 30

4.5 Метод Рунге-Кутты-Мерсона четвертого порядка 33

5 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА ИНТЕГРИРОВАНИЯ НА ЯЗЫКЕ C++ В СРЕДЕ QT CREATOR 37

5.1 Среда разработки Qt Creator 37

5.2 Описание программы 38

5.3 Графический интерфейс программы 38

5.4 Используемые библиотеки 39

5.5 Класс «Calc» с функциями вычисления ускорений 39

5.6 Основной код программы интегрирования 41
6 СРАВНИТЕЛЬНЫЙ АНАЛИЗ РЕЗУЛЬТАТОВ ЧИСЛЕННЫХ ИССЛЕДОВАНИЙ АЛГОРИТМОВ 51
ЗАКЛЮЧЕНИЕ 57
БИБЛИОГРАФИЧЕСКИЙ СПИСОК 58
ПРИЛОЖЕНИЕ А 59
ПРИЛОЖЕНИЕ Б 60
ПРИЛОЖЕНИЕ В 62
ПРИЛОЖЕНИЕ Г 81

ВВЕДЕНИЕ

Управление сложными техническими системами, к которым относятся и космические летательные аппараты (КЛА), основано на информации о состоянии объекта управления в текущий или заданный моменты времени. При этом информация о состоянии объекта должна быть получена с требуемой точностью и, как правило, за минимально возможное время.

Одним из основных параметров, характеризующих состояние КЛА, является его положение на орбите в определенные моменты времени. Прогнозирование движения КЛА в бортовых системах управления осуществляется для проведения коррекции орбиты, для приведения КЛА в заданную ориентацию с целью решения задач навигации или поставленных перед КЛА научных задач. В наземных системах управления полетом прогнозирование движения КЛА осуществляется для проведения устойчивых сеансов связи, раннего выявления возможных аварийных ситуаций и других задач.

Поэтому создание эффективных алгоритмов прогноза движения КЛА для систем управления ракетно-космическими комплексами разного вида является актуальной задачей. Следует так же отметить, что, в зависимости от цели решения задачи прогноза движения, требования к точности ее решения различны и составляют от нескольких метров до нескольких сотен метров.

Движение КЛА описывается системой дифференциальных уравнений, решение которой возможно только численными методами, в качестве которых обычно используются методы Рунге-Кутты различных порядков.

Целью моей ВКР является разработка методики сравнительной оценки применения методов Рунге-Кутты для решения задачи прогноза движения КЛА на участке свободного полета в гравитационном поле Земли без учета влияния атмосферы и проведение с ее использованием исследований для определения оптимального метода по критерию «заданная точность-минимальные вычислительные затраты».

Для достижения поставленной цели необходимо решить следующие задачи.

1) Провести анализ основных свойств движения КЛА на участке свободного полета и построить математическую модель движения.

2) Провести анализ методов Рунге-Кутты численного интегрирования дифференциальных уравнений и выбрать методы для проведения исследований.

3) На основе построенной математической модели и выбранных методов интегрирования разработать алгоритмы расчета параметров движения КЛА.

4) Разработать методику сравнительного анализа алгоритмов численного интегрирования уравнений движения КЛА.

5) Разработать программу, реализующую алгоритмы расчета параметров движения КЛА на участке свободного полета и обеспечивающую получение данных для их сравнительного анализа.

6) На основе сравнительного анализа выработать рекомендации по использованию алгоритмов расчета параметров движения на участке свободного полета в вычислительных комплексах бортовых и наземных систем управления полетом КЛА.

1 ПОСТРОЕНИЕ МАТЕМАТИЧЕСКОЙ МОДЕЛИ

1.1 Основные положения

Решение задач баллистики, как правило, проводится на основе математического моделирования движения ЛА. Состав моделей и их точность зависят как от содержания задачи, так и от этапа проектирования и разработки, на котором нужно решать задачу. Но в любом случае для построения таких моделей необходимо иметь математическое описание действующих на ЛА сил и процессов, которые происходят при полете. Кроме того, поскольку движение ЛА происходит в пространстве и времени, необходимо определить системы координат, в которых будут строиться математические модели. Совокупность подобных сведений в классической литературе по баллистике часто называют баллистической вычислительной моделью или, сокращенно, баллистической моделью, поскольку по умолчанию предполагается, что она предназначена для разработки программного обеспечения и проведения расчетов.

Основой баллистической модели ЛА является система дифференциальных уравнений его движения. Как правило, она не имеет аналитического решения. Его можно получить только методом численного интегрирования. Поэтому методы интегрирования, от выбора которых зависит точность решения задачи, также являются составной частью баллистической модели.

В зависимости от содержания задач, к точности расчета параметров траекторий предъявляются разные требования. Однако требование оптимизации вычислительных затрат остается практически постоянным. В связи с этим разработка оперативных алгоритмов расчета параметров движения ЛА, основанных на методах численного интегрирования системы дифференциальных уравнений, и определение зависимости точности решения от вычислительных затрат является актуальной и имеющей практическую ценность задачей.

В работе исследуется движение космического летательного аппарата (КЛА) на участке свободного полета (при отсутствии силы тяги двигательной установки) в гравитационном поле Земли.

В качестве модели Земли принят общеземной эллипсоид (эллипсоид вращения), центр которого совпадает с центром масс Земли. Гравитационное поле Земли (ГПЗ) соответствует гравитационному потенциалу общеземного эллипсоида (ОЗЭ) с точностью до квадрата сжатия. Такая модель ГПЗ обеспечивает требуемую точность при решении поставленной задачи.

В связи с тем, что атмосфера Земли оказывает влияние на движение КЛА на высотах до 80 км, а свободный участок полета проходит на высотах от 100 км и выше, полагаем, что влияние атмосферы на движение КЛА пренебрежимо мало.

Положения КЛА на орбите определяется положением его центра масс.

Движение КЛА будем рассматривать в неинерциальном пространстве, вращающемся вместе с Землей.

1.2 Уравнения движения КЛА в векторной форме

При принятых выше допущениях векторное уравнение движения центра масс КЛА в инерциальном пространстве имеет вид:

$$m\frac{d\vec{V}_a}{dt} = \vec{G} \qquad (1.1)$$

$\vec{G} = m\vec{g}$ – сила притяжения гравитационного поля Земли, где гравитационное ускорение $\vec{g}$ вычисляется по выбранной модели ГПЗ.

При анализе движения центра масс ЛА в неинерциальном пространстве, вращающемся вместе с Землей, следует учитывать переносные и кориолисовы силы инерции. В этом случае векторное уравнение движения имеет вид

$$m\frac{d\vec{V}}{dt} = \vec{G} - m\vec{a}_e - m\vec{a}_c \qquad (1.2)$$

здесь $\vec{a}_e = \vec{\omega}_3 \times [\vec{\omega}_3 \times \vec{r}]$ – переносное ускорение,

$\vec{a}_c = 2[\vec{\omega}_3 \times \vec{V}]$ – кориолисово ускорение.

$\vec{\omega}_3$ – угловая скорость вращения Земли.

$\vec{V}$ – скорость движения ЛА относительно вращающейся Земли.

В этом случае абсолютное ускорение ЛА вычисляется по формуле:

$$\frac{d\vec{V}_a}{dt} = \vec{a}_r + \vec{a}_e + \vec{a}_c \quad (1.3)$$

где $\vec{a}_r$ –ускорение ЛА относительно Земли.

В нашей баллистической модели уравнение (1.2) преобразуются к следующему виду:

$$\frac{d\vec{V}}{dt} = \vec{g} - \vec{a}_e - \vec{a}_c \quad (1.4)$$

1.3 Системы координат

Решение почти любой задачи баллистики ЛА включает выбор систем координат и их преобразование. Выбор системы координат зависит от постановки задачи и желания получить более простое решение, для сокращения времени и вычислительных затрат.

В практических задачах баллистики ЛА наибольшее распространение получили прямоугольные системы координат, которые определяются ортонормированными системами базисных векторов.

Прямоугольная система координат – система координат, оси которой перпендикулярны друг другу.

Геоцентрическая система координат – система координат, начало которой находится в центре масс Земли.

В наших расчетах будем использовать общеземную систему координат. Выбор этой системы координат связан с необходимостью определять положение КЛА относительно объектов, находящихся на Земле, например, наземных станций слежения и центра управления полетом КЛА

Общеземная система координат ($O\xi_{3C}\eta_{3C}\zeta_{3C}$) – это вращающаяся вместе с Землей, геоцентрическая, прямоугольная система координат с

началом в центре масс Земли. Ось $0\xi_{ЗС}$ направлена в точку пересечения Гринвичского меридиана с плоскостью Экватора, ось $0\zeta_{ЗС}$ направлена к Северному полюсу, ось $0\eta_{ЗС}$ дополняет систему до правой.

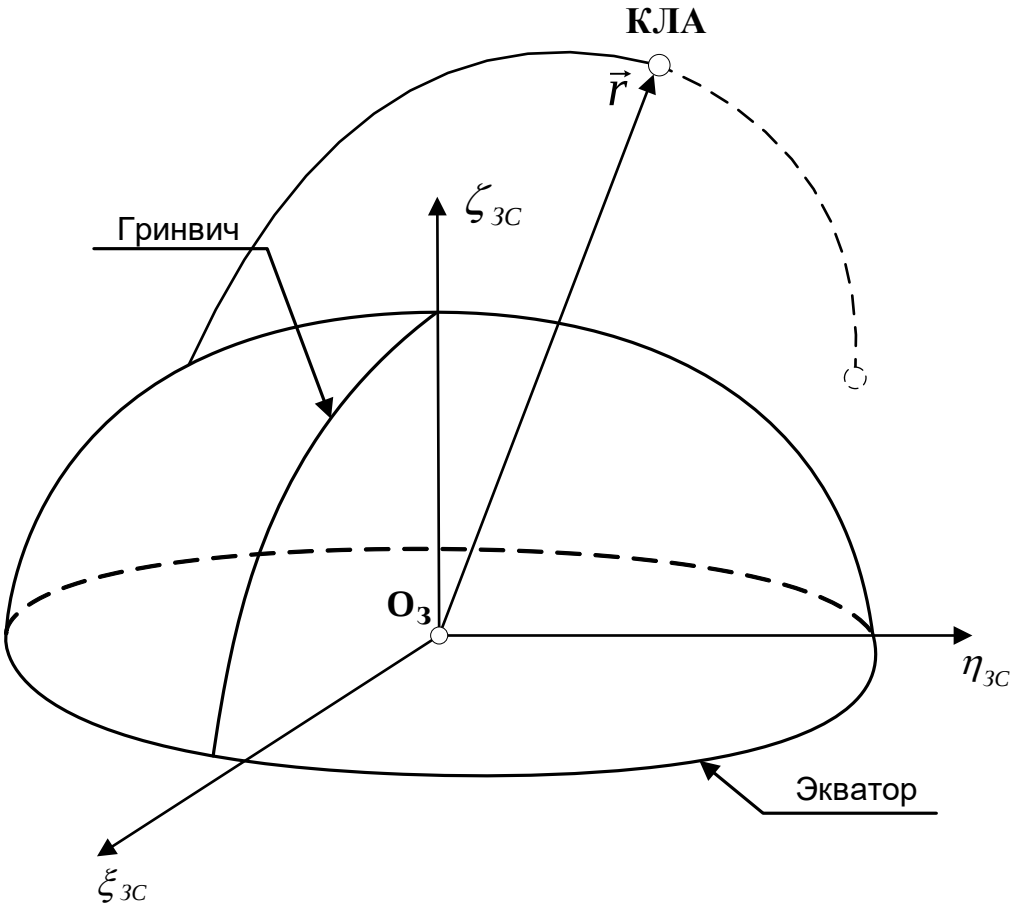


Рис. 1 Траектория движения КЛА в общеземной СК.

1.4 Система дифференциальных уравнений движения летательного аппарата

Векторное уравнение (1.4) является дифференциальным уравнением второго порядка, поскольку его правые части являются функциями не только скорости, но и координат центра масс КЛА. Поэтому приведем его к системе уравнений первого порядка.

$$\begin{cases} \frac{d\vec{V}_{\xi_{3C}}}{dt} = \begin{bmatrix} g_{\xi_{3C}} \\ g_{\eta_{3C}} \\ g_{\zeta_{3C}} \end{bmatrix} + \omega_з \begin{bmatrix} \omega_з \xi_{3C} + 2V_{\eta_{3C}} \\ \omega_з \eta_{3C} - 2V_{\xi_{3C}} \\ 0 \end{bmatrix} \\ \frac{d\vec{r}_{\xi_{3C}}}{dt} = \begin{bmatrix} V_{\xi_{3C}} \\ V_{\eta_{3C}} \\ V_{\zeta_{3C}} \end{bmatrix} \end{cases} \quad (1.5)$$

Начальные условия

$$\begin{bmatrix} \xi_{3C}^0 \\ \eta_{3C}^0 \\ \zeta_{3C}^0 \end{bmatrix}, \quad \begin{bmatrix} V_{\xi_{3C}}^0 \\ V_{\eta_{3C}}^0 \\ V_{\zeta_{3C}}^0 \end{bmatrix} \quad (1.6)$$

В нашей баллистической модели используется нормальное гравитационное поле Земли, соответствующее гравитационному потенциалу общеземного эллипсоида с точностью до квадрата сжатия. Такая модель ГПЗ обеспечивает требуемую точность при решении поставленной задачи.

Выражение для проекций вектора нормального гравитационного ускорения $\vec{g}$ на оси общеземной системы координат $O\xi_{3C}\eta_{3C}\zeta_{3C}$ имеет следующий вид:

$$\vec{g} = \begin{bmatrix} g_{\xi_{3C}} \\ g_{\eta_{3C}} \\ g_{\zeta_{3C}} \end{bmatrix} = \frac{1}{r} \begin{bmatrix} q \cdot \xi_{3C} \\ q \cdot \eta_{3C} \\ (q + \Delta q) \cdot \zeta_{3C} \end{bmatrix}, \quad (1.7)$$

где

$$r = \sqrt{\xi_{3C}^2 + \eta_{3C}^2 + \zeta_{3C}^2} \quad (1.8)$$

$$\mu = \left(\frac{\zeta_{3C}}{r}\right)^2 \quad (1.9)$$

$$q = -g_e \left(\frac{a_e}{r}\right)^2 \left[1 - \left(\frac{a_e}{r}\right)^2 \frac{3}{2} J_2 (3\mu - 1)\right] - \Delta q \cdot \mu \quad (1.10)$$

$$\Delta q = -3 g_e \left(\frac{a_e}{r}\right)^4 J_2 \quad (1.11)$$

Здесь

a_e – большая полуось общеземного эллипсоида,

g_e – гравитационное ускорение на Экваторе ОЗЭ,

J_2– коэффициент второй зональной гармоники,

r – расстояние от центра общеземного эллипсоида до точки внешнего пространства, в которой определяется значение нормального гравитационного ускорения.

2 ЧИСЛЕННЫЕ МЕТОДЫ РУНГЕ-КУТТЫ

Система дифференциальных уравнений движения центра масс ЛА не имеет аналитического решения. Его можно получить только методом численного интегрирования. Наибольшее распространение на практике для решения систем подобного вида получили методы Рунге-Кутты разных порядков.

Основная идея методов Рунге-Кутты заключается в том, что производные аппроксимируются через значения функции в точках на интервале, которые выбираются из условия наибольшей близости алгоритма к ряду Тейлора. В зависимости от старшей степени, с которой учитываются члены ряда, построены вычислительные схемы Рунге-Кутты разных порядков точности.

2.1 Метод Рунге-Кутты второго порядка

В 1895 году К. Рунге предложил при численном решении дифференциального уравнения

$$\frac{dy}{dx} = f(x, y(x)), y_n = y(x_n) \quad (2.1)$$

модифицировать метод Эйлера $y_{n+1} = y_n + h \cdot f(x_n, y_n)$ и использовать «правило средней точки» для вычисления значения y_{n+1} в следующем виде

$$k_1 = h \cdot f(x_n, y_n),$$

$$k_2 = h \cdot f\left(x_n + \frac{h}{2}, y_n + h\frac{k_1}{2}\right),$$

$$y_{n+1} = y_n + hk_2 \quad (2.2)$$

Идея умножения h на k_2 позволила существенно уменьшить погрешность численного решения (она ограничена величиной Ch^2).

2.2 Методы Рунге-Кутты высоких порядков

В 1901 году В. Кутта сформулировал общую схему того, что теперь называется методом Рунге-Кутты. Она базируется на двух следующих определениях.

Определение 1. Пусть s – целое положительное число («число стадий», или «этапов») и

$a_{21}, a_{31}, a_{32}, \dots, a_{s1}, a_{s1}, a_{s,s-1}, \dots, b_1, \dots, b_s, c_2, \dots, c_s$ – вещественные коэффициенты. Тогда метод

$$k_1 = h \cdot f(x_n, y_n),$$

$$k_2 = h \cdot f(x_n + c_2 h, y_n + h a_{21} k_1),$$

$$k_3 = h \cdot f(x_n + c_3 h, y_n + h(a_{31} k_1 + a_{32} k_2),$$

……..

$$k_s = h \cdot f\big(x_n + c_s h, y_n + h(a_{s1} k_1 + \dots + a_{s,s-1} k_{s-1})\big),$$

$$y_{n+1} = y_n + h(b_1 k_1 + \dots + b_s k_s) \qquad (2.3)$$

называется s –стадийным (s –этапным) явным методом Рунге-Кутта для задачи (5.1).

Обычно коэффициенты c_i удовлетворяют условиям

$$c_2 = a_{21}, c_3 = a_{31} + a_{32}, c_s = a_{s1} + \dots + a_{s,s-1}$$

$$c_i = \sum_j a_{ij} \qquad (2.4)$$

Смысл их в том, что все точки, в которых вычисляется f , являются приближениями первого порядка к решению. Эти условия сильно упрощают вывод условий, определяющих порядок аппроксимации для методов высокого порядка. Однако для методов низких порядков эти предположения не являются необходимыми.

Коэффициенты b_i удовлетворяют условию $\sum_i b_i = 1$. (2.5)

Определение 2. Метод Рунге-Кутта (5.3) имеет порядок p , если для достаточно гладких задач (5.1)

$$\|y(x_n + h) - y_{n+1}\| \leq Kh^{p+1} \quad (2.6)$$

т.е. если ряды Тейлора для точного решения $y(x_n + h)$ и для y_{n+1} совпадают до члена h^p включительно.

Из уравнений (2.4) и (2.5) следует, что для каждого порядка может быть получено несколько схем методов (наборов коэффициентов). При этом количество стадий (этапов) может не совпадать с порядком метода, который определяется по формуле (2.6).

При выполнении выпускной квалификационной работы, в проводимых исследованиях, кроме метода второго порядка (5.2), используются следующие методы:

3-х стадийный метод 3-го порядка:

$$k_1 = h \cdot f(x_n, y_n),$$
$$k_2 = h \cdot f\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right),$$
$$k_3 = h \cdot f(x_n + h, y_n + 2k_2 - k_1),$$
$$y_{n+1} = y_n + \frac{1}{6}h(k_1 + 4k_2 + k_3) \quad (2.7)$$

4-х стадийный метод 4-го порядка («Классический» метод)

$$k_1 = h \cdot f(x_n, y_n),$$
$$k_2 = h \cdot f\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right),$$
$$k_3 = h \cdot f\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right),$$
$$k_4 = h \cdot f(x_n + h, y_n + k_3),$$
$$y_{n+1} = y_n + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4) \quad (2.8)$$

5-ти стадийный метод 4-го порядка (Метод Рунге-Кутты-Мерсона)

$$k_1 = h \cdot f(x_n, y_n),$$
$$k_2 = h \cdot f\left(x_n + \frac{h}{3}, y_n + \frac{k_1}{3}\right),$$

$$k_3 = h \cdot f\left(x_n + \frac{h}{3}, y_n + \frac{1}{6}(k_1 + k_2)\right),$$

$$k_4 = h \cdot f\left(x_n + \frac{h}{2}, y_n + \frac{1}{8}(k_1 + 3k_3)\right) \quad (2.9)$$

$$\bar{y}_{n+1} = y_n + h\frac{1}{2}(k_1 - 3k_3 + 4k_4)$$

$$k_5 = h \cdot f(x_n + h, \bar{y}_{n+1}),$$

$$y_{n+1} = y_n + h\frac{1}{6}(k_1 + 4k_4 + k_5)$$

Отличие этого метода от классического метода Рунге-Кутты четвертого порядка состоит в том, что приближения точного решения $\bar{y}_{n+1}$и y_{n+1} с разными остаточными членами позволяют оценить погрешность численного метода, полученную на текущем шаге, и служат для повышения точности решения или (что бывает чаще) для автоматического выбора величины шага интегрирования.

3 МЕТОДИКА СРАВНИТЕЛЬНОГО АНАЛИЗА АЛГОРИТМОВ

Данная методика базируется на следующих основных положениях:

1. Расчет параметров орбиты КЛА на заданном временном интервале с использованием классического метода Рунге-Кутты 4-го порядка или метода Рунге-Кутты-Мерсона и шага интегрирования, равного1 с, может быть принят в качестве эталонного решения задачи. Это подтверждается практикой многочисленных расчетов.

2. Под точностью метода при расчете с шагом $h > 1c$ понимается отклонение положения КЛА в конце временного интервала от эталонного, которое вычисляется по формуле:

$$\varepsilon(h) = \sqrt{(\xi_{ЗСk}(h) - \xi_{ЗСk}(1))^2 + (\eta_{ЗСk}(h) - \eta_{ЗСk}(1))^2 + (\zeta_{ЗСk}(h) - \zeta_{ЗСk}(1))^2}$$

Таким образом, термины «точность» и «ошибка» метода в данном исследовании эквивалентны.

3. Поскольку основные временные затраты при интегрировании системы дифференциальных уравнений приходятся на вычисление правых частей, вычислительные затраты метода измеряются количеством вычислений правых частей в конкретном расчете $N_{ПЧ}(h)$.

Методика содержит следующие этапы:

1. Используя метод, Рунге-Кутты-Мерсона и шаг, равный 1 с, получить эталонное решение.

2. На заданной сетке шагов интегрирования в интервале [1,200] с для каждого исследуемого метода выполнить расчет параметров орбиты и вычислить отклонения положения КЛА от эталонного (ошибку метода) в зависимости от величины шага интегрирования $\varepsilon(h, метод)$. В процессе интегрирования считать количество вычислений правых частей системы уравнений (вычислительные затраты) $N_{ПЧ}(h, метод)$.

3. Используя зависимости $\varepsilon(h, метод)$ и $N_{ПЧ}(h, метод)$ построить зависимости $N_{ПЧ}(\varepsilon, метод)$.

4. Для заданного набора требуемых точностей решения задачи ε_{TP}^{j} из диапазона [1,200] м, используя зависимости $N_{ПЧ}(\varepsilon, метод)$ сформировать табличные функции $N_{ПЧ}^{j}(\varepsilon_{TP}^{j}, метод)$

5. Для каждого значения ε_{TP}^{j} провести сравнительный анализ методов и определить оптимальный по критерию «заданная точность-минимальные вычислительные затраты». Вычислить коэффициент эффективности оптимального метода по формуле

$$k_{ЭФ}^{j} = \frac{N_{ПЧ}^{j}(\varepsilon_{TP}^{j}, метод)}{N_{ПЧ}^{j}(\varepsilon_{TP}^{j}, опт.метод)} \quad (3.2)$$

4 АЛГОРИТМ ИНТЕГРИРОВАНИЯ МЕТОДАМИ РУНГЕ-КУТТЫ РАЗНЫХ ПОРЯДКОВ

Константы:

- большая полуось общеземного эллипсоида ($a_e = 6378136$, м);
- угловая скорость вращения Земли ($\omega_z = 0.00007292115$, м/с);
- коэффициент второй зональной гармоники нормального потенциала ($J_2 = 0.00108262575$);
- гравитационное ускорение на экваторе общеземного эллипсоида ($g_3 = 9.798288551648$, м/с^2).

Входные данные:

- проекции координат ($\xi_{3C0}, \eta_{3C0,}\varsigma_{3C0}$, м);
- проекции скоростей ($V_{\xi 3C0}, V_{\eta 3C0,}V_{\varsigma 3C0}$, м/с):
- проекции координат в момент времени t_k ($\xi_{3C}, \eta_{3C,}\varsigma_{3C}$, м);
- проекции скоростей в момент времени t_k ($V_{\xi 3C}, V_{\eta 3C,}V_{\varsigma 3C}$, м/с);
- начальный шаг интегрирования (h, м);
- конечный шаг интегрирования (h_e, м);
- время начала интегрирования (t_n, с);
- текущий момент интегрирования (t_k, c);
- конечный момент интегрирования (t_e, c);
- текущий момент времени полета ЛА (t_k, c).

Выходные данные:

- количество вычислений правых частей на участке интегрирования (N);
- величина ошибки интегрирования по координатам (er_r, м).

Для вычисления правых частей ДУ разработаны следующие функции.

Функция 1. Вычисление проекций гравитационного ускорения:

$$r = \sqrt{\xi_{3C}^2(t_k) + \eta_{3C}^2(t_k) + \zeta_{3C}^2(t_k)}$$

$$\mu = \left(\frac{\zeta_{3C}(t_k)}{r}\right)^2$$

$$\Delta q = -3g_e\left(\frac{a_e}{r}\right)^4 J_2$$

$$q = -g_e\left(\frac{a_e}{r}\right)^2\left[1 - \left(\frac{a_e}{r}\right)^2\frac{3}{2}J_2(3\mu - 1)\right] - \Delta q \cdot \mu$$

$$g_{\xi_{3C}}(t_k) = \frac{q}{r}\xi_{3C}(t_k)$$

$$g_{\eta_{3C}}(t_k) = \frac{q}{r}\eta_{3C}(t_k)$$

$$g_{\zeta_{3C}}(t_k) = \frac{(q+\Delta q)}{r}\zeta_{3C}(t_k)$$

Функция 2. Вычисление проекций переносного и кориолисова ускорений:

$$\dot{w}_{\xi_{3C}}^{ec}(t_k) = \omega_з(\omega_з\xi_{3C}(t_k) + 2V_{\eta_{3C}}(t_k))$$

$$\dot{w}_{\eta_{3C}}^{ec}(t_k) = \omega_з(\omega_з\eta_{3C}(t_k) - 2V_{\xi_{3C}}(t_k))$$

$$\dot{w}_{\zeta_{3C}}^{ec}(t_k) = 0$$

Функция 3. Расчет величины ошибки метода интегрирования производится по формуле:

$$er\_r(h) = \sqrt{(\xi_{3C}(h) - \xi_{3C}(1))^2 + (\eta_{3C}(h) - \eta_{3C}(1))^2 + (\zeta_{3C}(h) - \zeta_{3C}(1))^2}$$

Обозначим проекции координат $\xi_{3C0}, \eta_{3C0,}\varsigma_{3C0}$ как rn и проекции скоростей $V_{\xi 3C0}, V_{\eta 3C0,}V_{\varsigma 3C0}$ как vn, аналогично $\xi_{3C}(t_k), \eta_{3C}(t_k), \varsigma_{3C}(t_k)$ как rm, $V_{\xi 3C}(t_k), V_{\eta 3C}(t_k), V_{\varsigma 3C}(t_k)$ как vm.

Введем обозначение трех производных проекций координат и трех скоростей как fqr и fqv.

Обозначим проекции гравитационного ускорения $g_{\xi_{3C}}(t_k), g_{\eta_{3C}}(t_k), g_{\zeta_{3C}}(t_k)$ как g, а также проекций переносного и кориолисова ускорений $\dot{w}^{ec}_{\xi_{3C}}(t_k), \dot{w}^{ec}_{\eta_{3C}}, \dot{w}^{ec}_{\zeta_{3C}}$ как w.

Цикл 1. Повторяем до тех пор, пока не выполнится $h = h_e$ (не будет выполнено интегрирование со всеми заданными шагами от h до h_e.)

Цикл 2. Повторяем до тех пор, пока не выполнится условие $t_k = t_e$ увеличивая t_k по формуле $t_k = t_k + h$ после каждого шага.

Условие 1. Если шаг h будет больше, чем значение $t_e - t_k$, когда мы выйдем за правую границу времени, то изменяем шаг по формуле $h = t_e - t_k$, чтобы избежать этого.

4.1 Метод Рунге-Кутты-Мерсона с шагом h = 1

Расчеты этого метода с шагом равным единице будем считать точным решением. Он потребуется для вычисления ошибки интегрирования.

Сбрасываем счетчик $N = 0$

Начинаем интегрирование в цикле 2.

Проверка на условие 1.

Запоминаем начальное значение шага.

Этап 1

Присваиваем значения проекций в начале шага:

$rm = rn$

$vm = vn$

Вычислить проекции гравитационного ускорения

Функция 1(вход: rm, выход: g).

Вычислить проекции суммарного переносного и кориолисово ускорений

Функция 2(вход: rm, vm, выход: w)

Запоминаем проекции производных:

$fqr1 = vn$

$fqv1 = g + w$

Этап 2

$rm = rn + h * (fqr1)/3$

$vm = vn + h * (fqv1)/3$

Вычислить проекции гравитационного ускорения

Функция 1(вход: rm, выход: g).

Вычислить проекции суммарного переносного и кориолисового ускорений

Функция 2(вход: rm, vm, выход: w)

Запоминаем проекции производных:

$fqr2 = vm$

$fqv2 = g + w$

Этап 3

$rm = rn + h * (fqr1 + fqr2)/6$

$vm = vn + h * (fqv1 + fqv2)/6$

Передаём в функцию 1 значения rm, получаем g.

Передаём в функцию 2 значения rm и vm получаем w.

Запоминаем проекции производных:

$fqr3 = vm$

$fqv3 = g + w$

Этап 4

$rm = rn + h * (fqr1 + 3 * fqr3)/8$

$vm = vn + h * (fqv1 + 3 * fqv3)/8$

Вычислить проекции гравитационного ускорения

Функция 1(вход: rm, выход: g).

Вычислить проекции суммарного переносного и кориолисового ускорений

Функция 2(вход: rm, vm, выход: w)

Запоминаем проекции производных:

$fqr4 = vm$

$fqv4 = g + w$

Этап 4.5

$fqr4.5 = rn + h * (fqr1 - 3 * fqr3 + 4 * fqr4)/2$

$fqv4.5 = vn + h * (fqv1 - 3 * fqv3 + 4 * fqv4)/2$

Этап 5

$rm = fqr4.5$

$vm = fqv4.5$

Вычислить проекции гравитационного ускорения

Функция 1(вход: rm, выход: g).

Вычислить проекции суммарного переносного и кориолисового ускорений

Функция 2(вход: rm, vm, выход: w)

Запоминаем проекции производных:

$fqr5 = vm$

$fqv5 = g + w$

$rn = rn + h * (fqr1 + 4 * fqr4 + fqr5)/6$

$vn = vn + h * (fqv1 + 4 * fqv4 + fqv5)/6$

Увеличиваем значение счетчика на 5:

$N = N + 5$

Конец цикла 2.

Запоминаем точное решение для последующего расчета ошибки:

$er1 = rn$

$ev1 = vn$

4.2 Метод Рунге-Кутты второго порядка.

Начинаем интегрирование в цикле 1.

Сбрасываем счетчик $N = 0$.

Присваиваем в rn, vn начальные условия и $t_k = 0$.

Цикл 2.

Проверка на условие 1.

Запоминаем начальное значение шага.

Этап 1.

Присваиваем значения проекций в начале шага:

$rm = rn$

$vm = vn$

Вычислить проекции гравитационного ускорения

Функция 1(вход: rm, выход: g).

Вычислить проекции суммарного переносного и кориолисового ускорений

Функция 2(вход: rm, vm, выход: w)

Запоминаем проекции производных:

$fqr1 = vn$

$fqv1 = g + w$

Этап 2.

$rm = rn + 0.5 * h * fqr1$

$vm = vn + 0.5 * h * fqv1$

Вычислить проекции гравитационного ускорения

Функция 1(вход: rm, выход: g).

Вычислить проекции суммарного переносного и кориолисового ускорений

Функция 2(вход: rm, vm, выход: w)

Запоминаем проекции производных:

$fqr2 = vn$

$fqv2 = g + w$

Находим параметры движения в конце шага:

$rn = rn + h * fqr2$

$vn = vn + h * fqv2$

Увеличиваем значение счетчика на 2:

$N = N + 2$

Конец цикла 2.

Сохраняем точное решение для последующего расчета ошибки:

$er = rn$

$ev = vn$

Функция 3. (вход: er, ev, выход: er_r, er_v)

Получение данные сохраняем в файл для дальнейшего анализа.

Конец цикла 1.

4.3 Метод Рунге-Кутты третьего порядка

Начинаем интегрирование в цикле 1.

Сбрасываем счетчик $N = 0$.

Присваиваем в rn, vn начальные условия и $t_k = 0$.

Цикл 2.

Проверка на условие 1.

Запоминаем начальное значение шага.

Этап 1.

Присваиваем значения проекций в начале шага:

$rm = rn$

$vm = vn$

Вычислить проекции гравитационного ускорения

Функция 1(вход: $rm,$ выход: g).

Вычислить проекции суммарного переносного и кориолисового ускорений

Функция 2(вход: rm, vm, выход: w)

Запоминаем проекции производных:

$fqr1 = vn$

$fqv1 = g + w$

Этап 2.

$rm = rn + 0.5 * h * fqr1$

$vm = vn + 0.5 * h * fqv1$

Вычислить проекции гравитационного ускорения

Функция 1(вход: rm, выход: g).

Вычислить проекции суммарного переносного и кориолисового ускорений

Функция 2(вход: rm, vm, выход: w)

Запоминаем проекции производных:

$fqr2 = vn$

$fqv2 = g + w$

Этап 3.

$rm = rn + h * (2 * fqr2 - fqr1)$

$vm = vn + h * (2 * fqv2 - fqv1)$

Вычислить проекции гравитационного ускорения

Функция 1(вход: rm, выход: g).

Вычислить проекции суммарного переносного и кориолисового ускорений

Функция 2(вход: rm, vm, выход: w)

Запоминаем проекции производных:

$$fqr3 = vn$$

$$fqv3 = g + w$$

Находим параметры движения в конце шага:

$$rn = rn + h * (fqr1 + 4 * fqr2 + fqr3)/6$$

$$vn = vn + h * (fqv1 + 4 * fqv2 + fqv3)/6$$

Увеличиваем значение счетчика на 3:

$$N = N + 3$$

Конец цикла 2.

Сохраняем точное решение для последующего расчета ошибки:

$$er = rn$$

$$ev = vn$$

Функция 3. (вход: er, ev, выход: er_r, er_v)

Получение данные сохраняем в файл для дальнейшего анализа.

Конец цикла 1.

4.4 Метод Рунге-Кутты четвертого порядка

Начинаем интегрирование в цикле 1.

Сбрасываем счетчик $N = 0$.

Присваиваем в rn, vn начальные условия и $t_k = 0$.

Цикл 2.

Каждый шаг проверяем на условие 1. Также запоминаем начальное значение шага, для корректного расчета ошибки.

Этап 1.

Присваиваем значения проекций в начале шага:

$rm = rn$

$vm = vn$

Вычислить проекции гравитационного ускорения

Функция 1(вход: $rm,$ выход: g).

Вычислить проекции суммарного переносного и кориолисового ускорений

Функция 2(вход: $rm, vm,$ выход: w)

Запоминаем проекции производных:

$fqr1 = vn$

$fqv1 = g + w$

Этап 2.

$rm = rn + 0.5 * h * fqr1$

$vm = vn + 0.5 * h * fqv1$

Вычислить проекции гравитационного ускорения

Функция 1(вход: $rm,$ выход: g).

Вычислить проекции суммарного переносного и кориолисового ускорений

Функция 2(вход: $rm, vm,$ выход: w)

Запоминаем проекции производных:

$fqr2 = vn$

$fqv2 = g + w$

Этап 3.

$rm = rn + 0.5 * h * fqr2$

$vm = vn + 0.5 * h * fqv2$

Вычислить проекции гравитационного ускорения

Функция 1(вход: rm, выход: g).

Вычислить проекции суммарного переносного и кориолисового ускорений

Функция 2(вход: rm, vm, выход: w)

Запоминаем проекции производных:

$fqr3 = vn$

$fqv3 = g + w$

Этап 4.

$rm = rn + h * fqr3$

$vm = vn + h * fqv3$

Вычислить проекции гравитационного ускорения

Функция 1(вход: rm, выход: g).

Вычислить проекции суммарного переносного и кориолисового ускорений

Функция 2(вход: rm, vm, выход: w)

Запоминаем проекции производных:

$fqr4 = vn$

$fqv4 = g + w$

Находим параметры движения в конце шага:

$rn = rn + h * (fqr1 + 2 * (fqr2 + fqr3) + fqr4)/6$

$vn = vn + h * (fqv1 + 2 * (fqv2 + fqv3) + fqv4)/6$

Увеличиваем значение счетчика на 4:

$N = N + 4$

Конец цикла 2.

Сохраняем точное решение для последующего расчета ошибки:

$er = rn$

$ev = vn$

Функция 3. (вход: er, ev, выход: er_r, er_v)

Получение данные сохраняем в файл для дальнейшего анализа.

Конец цикла 1.

4.5 Метод Рунге-Кутты-Мерсона четвертого порядка

Начинаем интегрирование в цикле 1.

Сбрасываем счетчик $N = 0$.

Присваиваем в rn, vn начальные условия и $t_k = 0$.

Цикл 2.

Проверка на условие 1.

Запоминаем начальное значение шага.

Этап 1

Присваиваем значения проекций в начале шага:

$rm = rn$

$vm = vn$

Вычислить проекции гравитационного ускорения

Функция 1(вход: rm, выход: g).

Вычислить проекции суммарного переносного и кориолисового ускорений

Функция 2(вход: rm, vm, выход: w)

Запоминаем проекции производных:

$fqr1 = vn$

$fqv1 = g + w$

Этап 2

$rm = rn + h * (fqr1)/3$

$vm = vn + h * (fqv1)/3$

Вычислить проекции гравитационного ускорения

Функция 1(вход: rm, выход: g).

Вычислить проекции суммарного переносного и кориолисового ускорений

Функция 2(вход: rm, vm, выход: w).

Запоминаем проекции производных:

$fqr2 = vm$

$fqv2 = g + w$

Этап 3

$rm = rn + h * (fqr1 + fqr2)/6$

$vm = vn + h * (fqv1 + fqv2)/6$

Вычислить проекции гравитационного ускорения

Функция 1(вход: rm, выход: g).

Вычислить проекции суммарного переносного и кориолисового ускорений

Функция 2(вход: rm, vm, выход: w).

Запоминаем проекции производных:

$fqr3 = vm$

$fqv3 = g + w$

Этап 4

$rm = rn + h * (fqr1 + 3 * fqr3)/8$

$vm = vn + h * (fqv1 + 3 * fqv3)/8$

Вычислить проекции гравитационного ускорения

Функция 1(вход: rm, выход: g).

Вычислить проекции суммарного переносного и кориолисового ускорений

Функция 2(вход: rm, vm, выход: w).

Запоминаем проекции производных:

$fqr4 = vm$

$fqv4 = g + w$

Этап 4.5

$fqr4.5 = rn + h * (fqr1 - 3 * fqr3 + 4 * fqr4)/2$

$fqv4.5 = vn + h * (fqv1 - 3 * fqv3 + 4 * fqv4)/2$

Этап 5

$rm = fqr4.5$

$vm = fqv4.5$

Вычислить проекции гравитационного ускорения

Функция 1(вход: rm, выход: g).

Вычислить проекции суммарного переносного и кориолисового ускорений

Функция 2(вход: rm, vm, выход: w).

Запоминаем проекции производных:

$fqr5 = vm$

$fqv5 = g + w$

$rn = rn + h * (fqr1 + 4 * fqr4 + fqr5)/6$

$vn = vn + h * (fqv1 + 4 * fqv4 + fqv5)/6$

Увеличиваем значение счетчика на 5:

$N = N + 5$

Конец цикла 2.

Сохраняем точное решение для последующего расчета ошибки:

$er = rn$

$ev = vn$

Функция 3. (вход: er, ev, выход: er_r, er_v)

Получение данные сохраняем в виде файла для дальнейшего анализа.

Конец цикла 1. Конец алгоритма.

5 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА ИНТЕГРИРОВАНИЯ НА ЯЗЫКЕ C++ В СРЕДЕ QT CREATOR

5.1 Среда разработки Qt Creator

Qt Creator (ранее назывался Greenhouse) — кроссплатформенная свободная IDE для разработки на таких языках как C, C++ и QML. Данная среда разработана на основе Trolltech, предназначена для работы с фреймворком Qt. В него входят графический интерфейс отладчика и визуальные средства разработки интерфейса как с использованием QtWidgets, так и QML.

Основной задачей Qt Creator является упрощение разработки приложения под разные платформы используя фреймворк Qt. С использованием данного фреймворка было написано много популярных программ и приложений: 2ГИС для Android, Kaspersky Internet Security, Virtual Box, Skype, VLC Media Player, Opera, Сетевая карта мира Google Earth и другие.

Qt использует MOC (Meta Object Compiler) для предварительной компиляции программ. Исходный текст программы обрабатывается MOC, который ищет в классах программы макрос Q_OBJECT и переводит исходный код в мета-объектный код, после чего мета-объектный код компилируется компилятором C++. MOC расширяет функциональность фреймворка, благодаря ему добавляются такие понятия, как слоты и сигналы.

В данной среде реализовано автодополнение, включающее в том числе ключевые слова стандарта C++11 (с версии 2.5), подцветка текста по собственному желанию или по стандартным шаблонам самой среды.

Его основным преимуществом является кроссплатформенность, что значительно ускоряет создание приложений для разных операционных систем, таких как Windows, Mac OS, Linix, а также мобильных и встроенных платформ. Также нельзя не упомянуть о простом и удобном создании интерфейса программы в Qt Designer.

5.2 Описание программы

Разработанная программа предназначена для интегрирования уравнений движения летательного аппарата. Реализация данной программы происходило на языке C++ в среде программирования «QT», которые оказались более удобными для реализации данной выпускной квалификационной работы.

5.3 Графический интерфейс программы

Для корректной работы необходимо занести начальные условия интегрирования: временной участок, на котором будет происходить интегрирование (от t_n до t_e), диапазон шага (от h_n до h_e с интервалом dh), а также проекции координат и скоростей в начальный момент времени. После заполнения всех полей нажимаем кнопку «Расчет». На рисунке 2 изображен внешний вид программы при первом запуске. Также туда будут занесены начальные условия интегрирования.

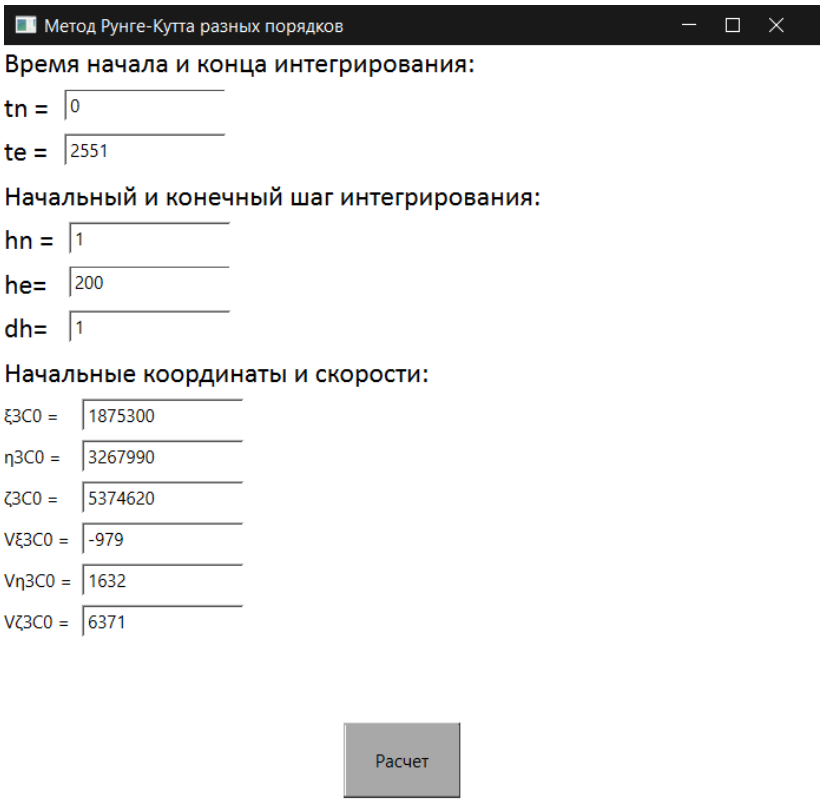


Рис.2 Графический интерфейс программы

5.4 Используемые библиотеки

#include - директива для указывающая препроцессору подключать файлы библиотек, где храниться описание функций и т.д.

Класс <QMainWindow> — это главное окно, которое предоставляет структуру для создания пользовательского интерфейса приложения.

Класс <QDebug> используется всякий раз, когда разработчику необходимо записать отладочную или трассировочную информацию на устройство, файл, строку или консоль.

Класс <QFile> — это класс ввода-вывода для чтения и записи текстовых и двоичных файлов и ресурсов.

Класс <QtCore/qmath.h> предоставляет различные математические функции.

Модуль <QtCore> содержит основные функции, не связанные с графическим интерфейсом.

5.5 Класс «Calc» с функциями вычисления ускорений

В данном классе находятся функции расчета проекций гравитационного, кориолисово и переносного ускорений.

В заголовочном файле calc.h подключаются все необходимые библиотеки и модули и объявляется класс calc(). Внутри него объявляются константы, которые будут использоваться в функциях расчета ускорений. Такие, как:

- большая полуось общеземного эллипсоида ($a_e = 6378136$);
- угловая скорость вращения Земли ($\omega_z = 0.00007292115$);
- коэффициент второй зональной гармоники нормального потенциала ($J_2 = 0.00108262575$);

- гравитационное ускорение на экваторе общеземного эллипсоида ($g_3 = 9.798288551648$).

Также в этом файле объявляются сами функции типа double с плавающей запятой, для расчета ускорений.

double gravx(double x, double y, double z);

double gravy(double x, double y, double z);

double gravz(double x, double y, double z);

В файле calc.cpp происходит расчет проекций гравитационного ускорений происходит в функциях gravx, gravy, gravz. В дальнейшем мы будем обращаться к ним, передавать туда проекции координат ($\xi_{3C}, \eta_{3C,}\varsigma_{3C}$), обозначенные как double x, double y, double z соответственно. На выходе получим искомые ускорения в данный момент времени.

Аналогично с функциями вычисления переносного и кориолисово ускорений. Но их две, так как вертикальную ось проекция нулевая в следствии того, что вращение происходит вокруг данной оси и соответственно проекция равна нулю.

double korandperx(double x, double vy);

double korandpery(double y, double vx);

Для расчета проекций данных ускорений в данный момент времени мы обращаемся к данным функциям. В korandperx мы передаём ξ_{3C} на место переменной x и $V_{\eta 3C}$ на место переменной vy. В korandpery мы передаём $\eta_{3C,}$на место переменной x и $V_{\xi 3C}$ на место переменной vy. Код файла calc.cpp смотреть в Приложении Б.

5.6 Основной код программы интегрирования

В заголовочном файле mainwindow.h подключаются все необходимые библиотеки, файлы и модули, такие как QtCore/qmath.h, QMainWindow, QDebug, calc.h и QFile.

Также в нем объявляются все переменные типа double, которые будут задействованы во время работы программы.

Проекции координат в начальный (xn, yn, zn) и текущий момент времени (x, y, z).

```
double xn, yn, zn, x, y, z;
```

Шаг интегрирования (начальный h, конечный hk и переменная, с которой этот шаг будет меняться dh), а также время начала tn и конца интегрирования tk.

```
double h, hk, dh, tn, tk;
```

Проекции скоростей в начальный (vxn, vyn, vzn) и текущий момент времени (vx, vy, vz).

```
double vxn, vyn, vzn, vx, vy, vz;
```

Производные координат и скоростей на этапах интегрирования. Для метода второго порядка будут использованы переменные fq1vx, fq1vy, fq1vz, fq2vx, fq2vy, fq2vz для производных проекций координат и fq1ax, fq1ay, fq1az, fq2ax, fq2ay, fq2az для производных проекций скоростей. Аналогично для более высоких порядков.

```
double fq1ax, fq1ay, fq1az, fq1vx, fq1vy, fq1vz, fq2ax, fq2ay, fq2az, fq2vx, fq2vy, fq2vz;

double fq3ax, fq3ay, fq3az, fq3vx, fq3vy, fq3vz, fq4ax, fq4ay, fq4az, fq4vx, fq4vy, fq4vz;
```

double fq45ax, fq45ay, fq45az, fq45vx, fq45vy, fq45vz, fq5ax, fq5ay, fq5az, fq5vx, fq5vy, fq5vz;

Величина ошибок по координатам и скоростям для порядков метода от второго по пятый. Для метода второго порядка будут использованы переменные er_point2 для координат и er_speed2 для скоростей. Аналогично для более высоких порядков.

double er_point2, er_speed2, er_point3, er_speed3, er_point4, er_speed4, er_point5, er_speed5;

Переменные, в которых будут содержаться проекции координат и скоростей, по которым производится расчет ошибки.

double evx1, evy1, evz1, ex1, ey1, ez1, evx, evy, evz, ex, ey, ez;

Проекции гравитационного ускорения (gx, gy, gz), кориолисова и переносного (wzx, wzy) в текущий момент времени.

double gx, gy, gz, wzx, wzy;

Счетчик колличества расчета правых частей N.

double N;

Переменная, в которой запоминаем шаг h для корректной работы цикла.

double remember_h;

В этом же заголовочном файле объявляем функцию void on_result_released(). В момент, когда мы отжимаем кнопку «calc» в интерфейсе программы выполняется данная функция, которая и производит интегрирование методами Рунге-Кутты второго-четвертого порядков.

Все основные расчеты, в том числе интегрирование находятся в файле mainwindow.cpp.

Первым делом подключаем заголовочный файл и файл графического интерфейса для корректной работы программы.

Объявляем функцию void MainWindow::on_result_released(). При начатии на кнопку в графическом интерфейсе будет выполняться данная функция. Затем создаём локальную переменную w для работы с классом calc.

calc* w = new calc;

Далее начинаем работу с файлами. Переменной p2 присваиваем файл, который расположен по адресу C:\\1integration_result\\errorpoints2.txt. В него будет сохранять ошибки интегрирования метода Рунге-Кутты второго порядка по координатам.

Переменной n2 присваиваем файл, который находится по адресу C:\\1integration_result\\N2.txt и в этот файл поместим количество расчетов метода Рунге-Кутты второго порядка.

Аналогично для методов третьего, четвертого и четвертого Мерсона порядков

```
QFile p2("C:\\1integration_result\\errorpoints2.txt");
QFile p3("C:\\1integration_result\\errorpoints3.txt");
QFile p4("C:\\1integration_result\\errorpoints4.txt");
QFile p5("C:\\1integration_result\\errorpoints5.txt");
QFile n2("C:\\1integration_result\\N2.txt");
QFile n3("C:\\1integration_result\\N3.txt");
QFile n4("C:\\1integration_result\\N4.txt");
QFile n5("C:\\1integration_result\\N5.txt");
```

Затем открываем эти файлы и указываем, что мы не будем считывать информацию из них, будет осуществляться только запись.

```
p2.open(QIODevice::WriteOnly);

p3.open(QIODevice::WriteOnly);

p4.open(QIODevice::WriteOnly);

p5.open(QIODevice::WriteOnly);

n2.open(QIODevice::WriteOnly);

n3.open(QIODevice::WriteOnly);

n4.open(QIODevice::WriteOnly);

n5.open(QIODevice::WriteOnly);
```

Начинаем интегрирование. Начнем с метода Рунге-Кутты-Мерсона четвертого порядка для расчета нашего точного решения.

Задаем шаг интегрирования равным единице.

```
h = 1;
```

Получаем значения времени начала и конца интегрирования, начальные координаты и начальные скорости из ui (user interface).

```
tn = ui->tn->text().toDouble();

tk = ui->tk->text().toDouble();

xn = ui->xn->text().toDouble();

yn = ui->yn->text().toDouble();

zn = ui->zn->text().toDouble();

vxn = ui->vxn->text().toDouble();

vyn = ui->vyn->text().toDouble();

vzn = ui->vzn->text().toDouble();
```

В конце каждой строки стоит метод toDouble(), который преобразует значение заданного объекта в число с плавающей запятой двойной точности.

После того, как мы получили все необходимые данные из расчетов, можно приступать к интегрированию правых частей уравнений движения.

В цикле for по времени, который пробегает значения от tn (начальный момент времени до tk (конечный момент времени) постепенно увеличивая значение tn на величину tn+h, те к предыдущему значению времени прибавляется шаг h, в данном случае он равен единице. Данный цикл будет повторяться, пока выполняется условие tn<tk.

for (tn; tn<tk; tn=tn+h){…

Этап 1. Присваиваем начальные значения переменным, через которые будем производить интегрирование.

```
x = xn;       vx = vxn;

y = yn;       vy = vyn;

z = zn;       vz = vzn;
```

Обращаемся к классу calc с помощью переменной w, которую объявили ранее. Далее через функцию gravx вычисляем проекцию гравитационного ускорения на первую ось (отправляем в функцию значения x, y, z) и присваиваем полученное значение в переменную gx. Аналогично на две другие оси.

```
gx = w->gravx(x, y, z);

gy = w->gravy(x, y, z);

gz = w->gravz(x, y, z);
```

Таким же способом получаем проекции кориолисова и переносного ускорения на первую и вторую ось (на третью ось проекция равна нулю) В функцию отправляем переменные, которые указаны в скобках в коде ниже.

wzx = w->korandperx(x,vy);

wzy = w->korandpery(y,vx);

Необходимо сохранить производные координат и скоростей для дальнейших расчетов в заранее объявленные для этого переменные.

fq1vx = vxn; fq1ax = gx+wzx;

fq1vy = vyn; fq1ay = gy+wzy;

fq1vz = vzn; fq1az = gz;

Этап 2. Рассчитываем

Находим промежуточные значения проекций на шаге $\frac{h}{3}$.

x = xn + h*fq1vx/3;

y = yn + h*fq1vy/3;

z = zn + h*fq1vz/3;

vx = vxn + h*fq1ax/3;

vy = vyn + h*fq1ay/3;

vz = vzn + h*fq1az/3;

Через функцию gravx() и korandperx() вычисляем проекции гравитационного ускорения и кориолисова с переносным используя полученые значения x, y, z, vx, vy, vz.

Сохраняем производные координат и скоростей для дальнейших расчетов.

fq2vx = vxn; fq2ax = gx+wzx;

fq2vy = vyn; fq2ay = gy+wzy;

fq2vz = vzn; fq2az = gz;

Этап 3.

Находим промежуточные значения проекций по следующим формулам:

x = xn + h*(fq1vx+fq2vx)/6;

y = yn + h*(fq1vy+fq2vy)/6;

z = zn + h*(fq1vz+fq2vz)/6;

vx = vxn + h*(fq1ax+fq2ax)/6;

vy = vyn + h*(fq1ay+fq2ay)/6;

vz = vzn + h*(fq1az+fq2az)/6;

С помощью функций gravx() и korandperx() вычисляем проекции гравитационного ускорения и кориолисова с переносным используя полученные значения x, y, z, vx, vy, vz.

Сохраняем производные координат и скоростей для дальнейших расчетов.

fq3vx = vxn; fq3ax = gx+wzx;

fq3vy = vyn; fq3ay = gy+wzy;

fq3vz = vzn; fq3az = gz;

Этап 4.

Находим промежуточные значения проекций по следующим формулам:

x = xn + h*(fq1vx+3*fq3vx)/8;

y = yn + h*(fq1vy+3*fq3vy)/8;

z = zn + h*(fq1vz+3*fq3vz)/8;

vx = vxn + h*(fq1ax+3*fq3ax)/8;

vy = vyn + h*(fq1ay+3*fq3ay)/8;

vz = vzn + h*(fq1az+3*fq3az)/8;

С помощью функций gravx() и korandperx() вычисляем проекции гравитационного ускорения и кориолисова с переносным используя полученные значения x, y, z, vx, vy, vz.

Сохраняем производные координат и скоростей для дальнейших расчетов.

fq4vx = vx; fq4ax = gx+wzx;

fq4vy = vy; fq4ay = gy+wzy;

fq4vz = vz; fq4az = gz;

Этап 4.5

Находим промежуточные значения проекций по следующим формулам и сразу запоминаем их:

fq45vx = xn + h*(fq1vx-3*fq3vx+4*fq4vx)/2;

fq45vy = yn + h*(fq1vy-3*fq3vy+4*fq4vy)/2;

fq45vz = zn + h*(fq1vz-3*fq3vz+4*fq4vz)/2;

fq45ax = vxn + h*(fq1ax-3*fq3ax+4*fq4ax)/2;

fq45ay = vyn + h*(fq1ay-3*fq3ay+4*fq4ay)/2;

fq45az = vzn + h*(fq1az-3*fq3az+4*fq4az)/2;

Этап 5.

Присваиваем переменным x, y, z, vx, vy, vz значения, которые получили на этапе 4.5.

x = fq45vx;

y = fq45vy;

z = fq45vz;

vx = fq45ax;

vy = fq45ay;

vz = fq45az;

С помощью функций gravx() и korandperx() вычисляем проекции гравитационного ускорения и кориолисова с переносным используя полученные значения x, y, z, vx, vy, vz.

Сохраняем производные координат и скоростей.

fq5vx = vx; fq5ax = gx+wzx;

fq5vy = vy; fq5ay = gy+wzy;

fq5vz = vz; fq5az = gz;

Находим значения проекций скоростей и ускорений на конце шага h.

xn = xn + h*(fq1vx+4*fq4vx+fq5vx)/6;

yn = yn + h*(fq1vy+4*fq4vy+fq5vy)/6;

zn = zn + h*(fq1vz+4*fq4vz+fq5vz)/6;

vxn = vxn + h*(fq1ax+4*fq4ax+fq5ax)/6;

vyn = vyn + h*(fq1ay+4*fq4ay+fq5ay)/6;

vzn = vzn + h*(fq1az+4*fq4az+fq5az)/6;

Увеличиваем значения счетчика на пять $N = N + 5$.

Присваиваем переменным ex1, ey1, ez1 значения, которые получили в конце цикла.

Цикл повторяется, пока не выполнится условие tn<tk.

Конец расчета точного решения методом Рунге-Кутты-Мерсона четвертого порядка. Полученные значения будем использовать в дальнейших расчетах.

Программная реализация алгоритмов интегрирования исследуемыми методами Рунге-Кутты второго, третьего и четвертого порядка и Рунге-Кутты-Мерсона с переменным шагом выполнена следующим образом.

1. Из графического интерфейса получаем начальные данные.
2. Объявляем внешний цикл, который будет повторять расчет, пока не будут произведены расчеты с шагом, начиная от начального, до конечного.
3. Внутри этого цикла объявляется ещё один, в котором и происходит интегрирование согласно алгоритму. Также внутри этого цикла происходит проверка, чтобы мы не выходили за значение конечного промежутка времени, если шаг будет больше его. Если шаг больше необходимого, то его уменьшаем до необходимого нам.
4. В конце внешнего цикла мы рассчитываем ошибку интегрирования по координатам и скоростям, после чего сохраняем их в соответствующие файлы с точностью до трех знаков после запятой.
5. После выполнения внешнего цикла мы закрываем файлы, с которыми работали, так как они больше не будут использоваться.

Листинг программы приведен в Приложении В.

6 СРАВНИТЕЛЬНЫЙ АНАЛИЗ РЕЗУЛЬТАТОВ ЧИСЛЕННЫХ ИССЛЕДОВАНИЙ АЛГОРИТМОВ

В соответствии с методикой сравнительного анализа алгоритмов по разработанной программе в диапазоне шагов интегрирования $h \in [1,200]$ с дискретом 1 с для каждого исследуемого алгоритма были определены величины ошибок метода интегрирования $\varepsilon(h, \text{метод})$ и количество вычислений правых частей (вычислительные затраты) $N_{\text{Пч}}(h, \text{метод})$ от величины шага интегрирования.

Численные исследования были проведены для участка свободного полета КЛА с начальными условиями:

Границы интервала интегрирования, с

То=0 Тк=2551

Проекции координат, м

ξ =1875300

η =3267990

ζ=5374620

Проекции скоростей, м/с

V_ξ = -979

V_η = 1632

V_ζ = 6371

Сравнительный анализ проводился для следующих значений заданного набора требуемой точности решения задачи определения положения КЛА.

$\varepsilon_{\text{ТР}}$, м									
1	5	10	15	20	50	80	100	150	200

Результаты расчетов приведены в таблице Приложения Г и показаны на рисунках 3-8.

На рисунках обозначены:

РК2 – метод Рунге-Кутты 2-го порядка;

РК3 – метод Рунге-Кутты 3-го порядка;

РК4 – метод Рунге-Кутты 4-го порядка;

РК5 – пятистадийный метод Рунге-Кутты-Мерсона четвертого порядка.

На рисунках 3 и 4 показаны графические зависимости ошибки методов от величины шага интегрирования, построенные по результатам расчетов.

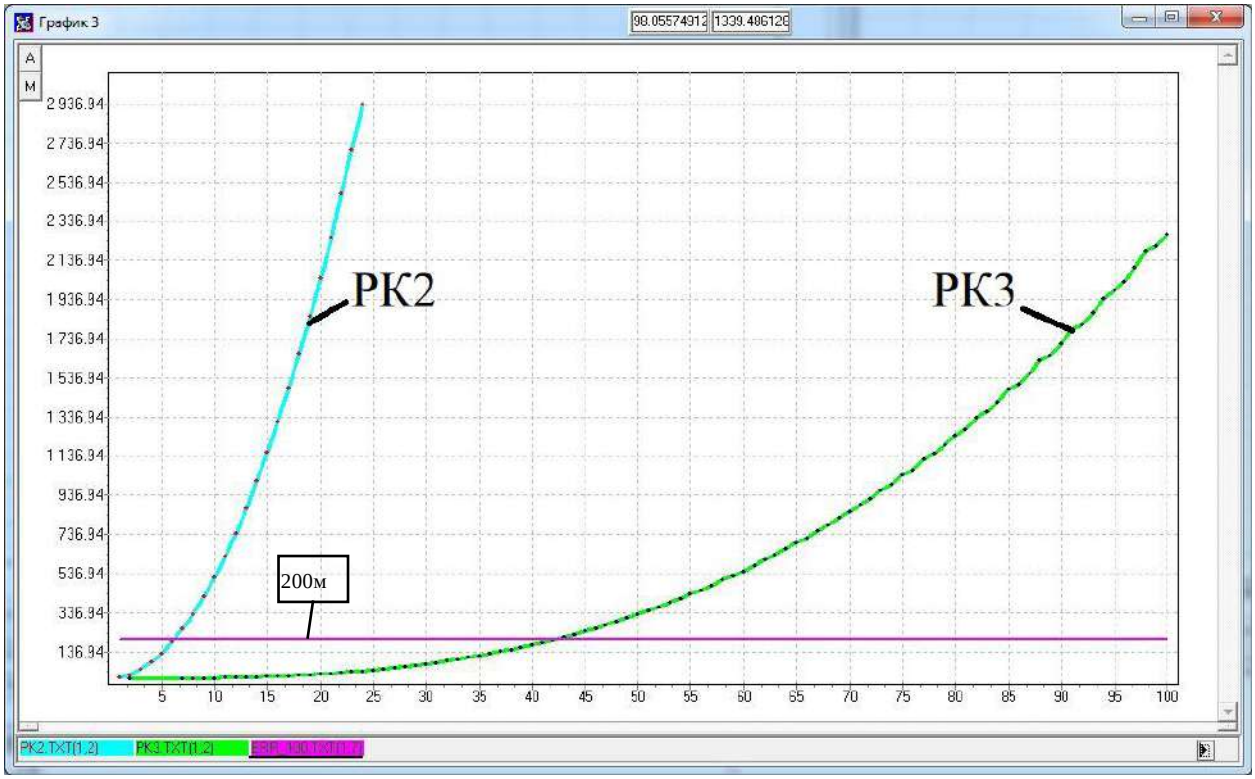


Рис. 3 Зависимость ошибки методов РК2 и РК3 от величины шага h.

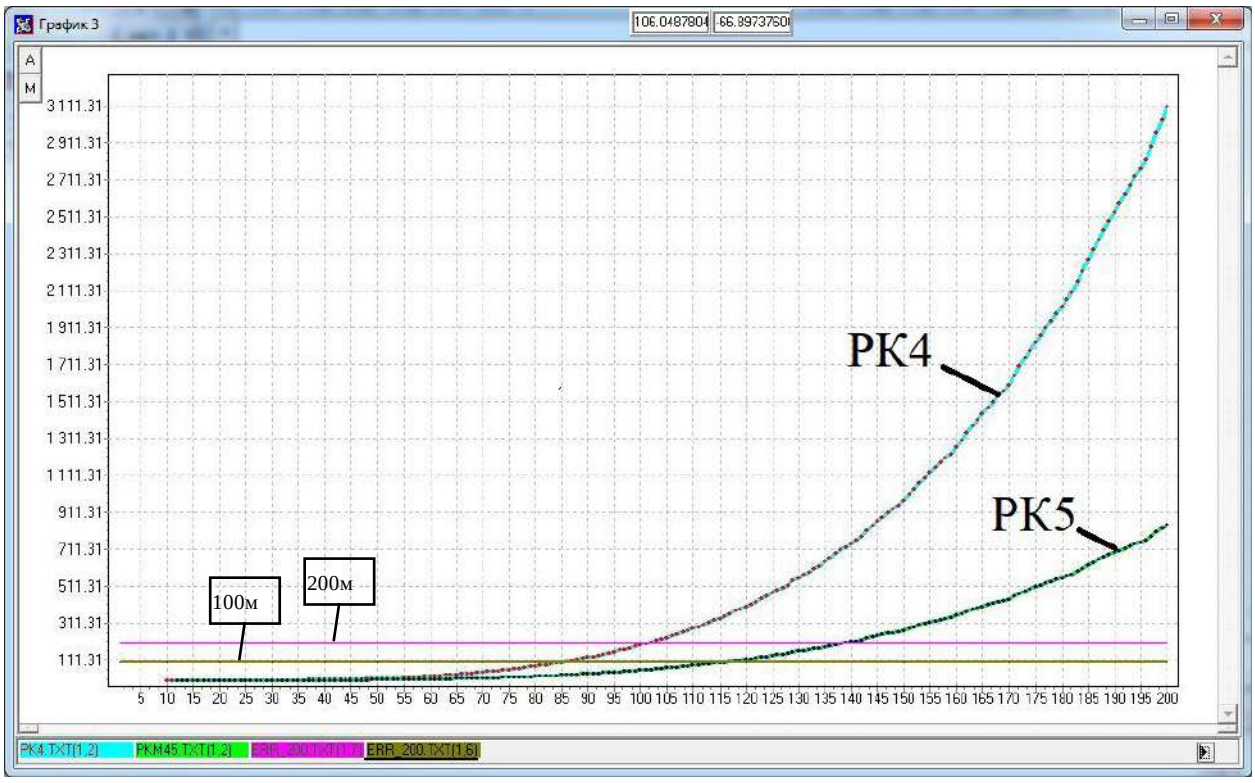


Рис. 4 Зависимость ошибки методов РК4 и РК5 от величины шага h.

Ошибки методов Рунге-Кутты 2-го и 3-го порядков оказались настолько велики, что эти методы были исключены как бесперспективные для дальнейших исследований. Например, для достижения заданной точности расчета, равной 5 м, методом Рунге-Кутты 2-го порядка шаг интегрирования должен быть равен 1 с, при этом количество вычислений правых частей равно 5102. Методом Рунге-Кутты 3-го порядка точность 5 м достигается при шаге, равном 12 с, при этом количество вычислений правых частей равно 639. Методы Рунге-Кутты 4-го порядка и Рунге-Кутты-Мерсона обеспечивают точность 5 м при шаге 40 с и 54 с, и количестве вычислений правых частей 256 и 240, соответственно. Заданная точность, равная 200 м, обеспечивается методами Рунге-Кутты 2-го порядка, и Рунге-Кутты 3-го порядка при шаге интегрирования 6 с и 42 с, при этом количество вычислений правых частей равно 852 и 183, соответственно. Методы Рунге-Кутты 4-го порядка и Рунге-Кутты-Мерсона обеспечивают точность 200 м при шаге 101 с и 138 с, и количестве вычислений правых частей 104 и 95.

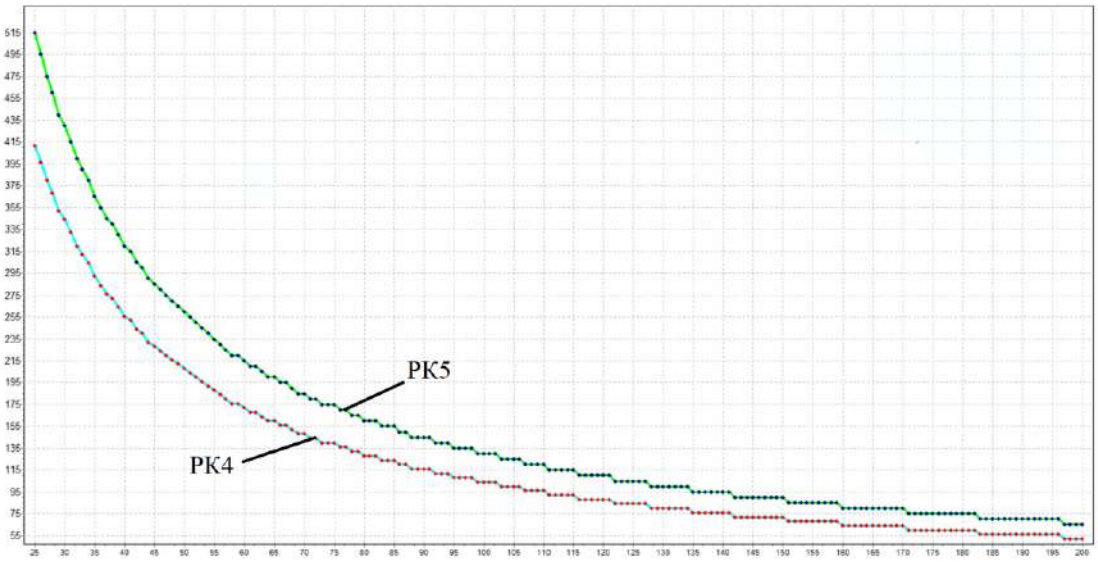
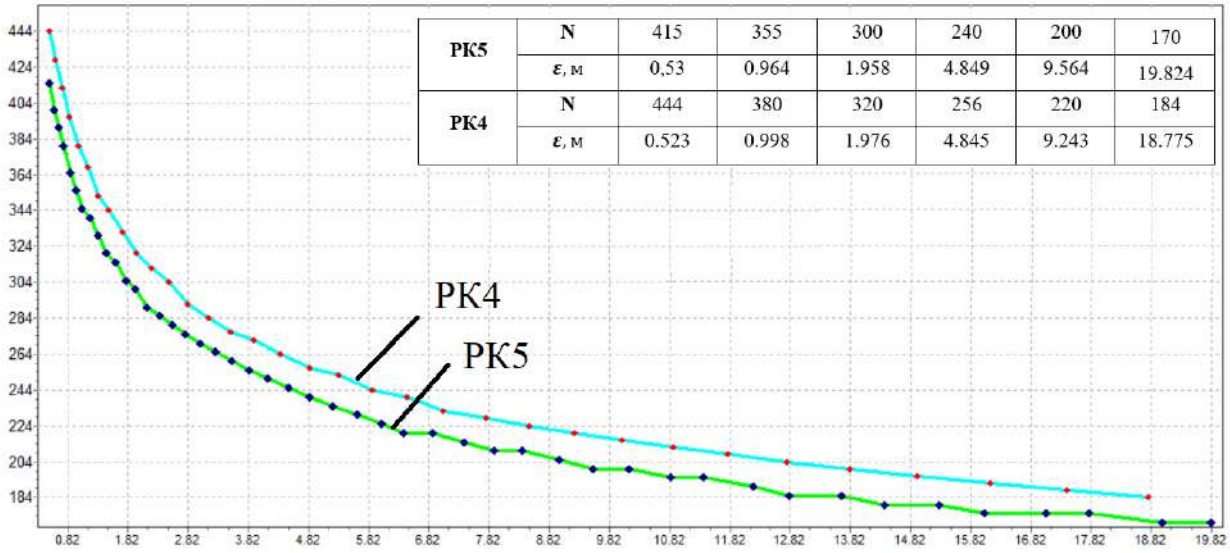


Рис. 5 Зависимость количества вычислений правых частей в методах РК4 и РК5 от величины шага h.

На рисунке 5 показаны зависимости количества вычислений правых частей от величины шага интегрирования классического метода Рунге-Кутты 4-го порядка и метода Рунге-Кутты-Мерсона. Здесь следует отметить быстрое уменьшение количества вычислений правых частей с увеличением шага на участке до 60 с и появление, начиная с шага ~70 с, участков на которых увеличение шага не приводит к увеличению количества вычисления правых частей. Эти факторы существенно влияют на результаты сравнительной оценки исследуемых методов.

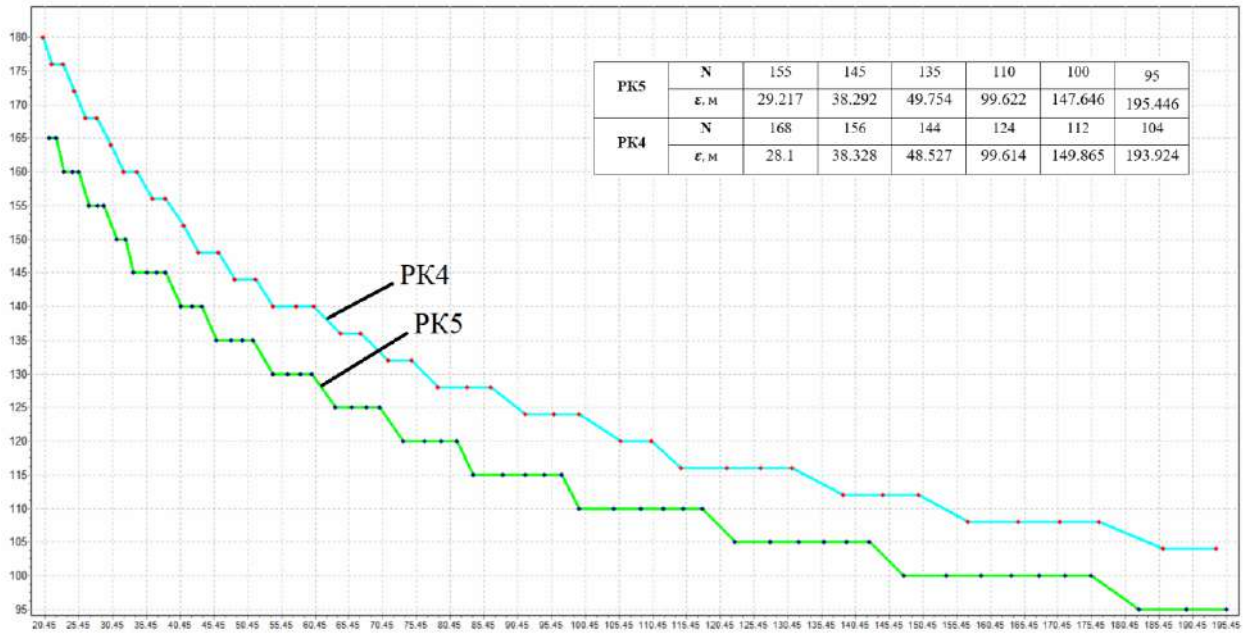
На рисунках 6 и 7 приведены табличные и графические зависимости ошибок методов от количества вычислений правых частей. Они получены на основе данных, приведенных на предыдущих графиках. Для удобства анализа общий диапазон ошибок методов [0.5,200] м разделен на два диапазона [0.5,20] м и [20,200] м.

Они показывают, что при одинаковых ошибках методов количество вычислений правых частей в методе Рунге-Кутты-Мерсона оказывается меньше во всем диапазоне.



РК5	N	415	355	300	240	200	170
	ε, м	0,53	0.964	1.958	4.849	9.564	19.824
РК4	N	444	380	320	256	220	184
	ε, м	0.523	0.998	1.976	4.845	9.243	18.775

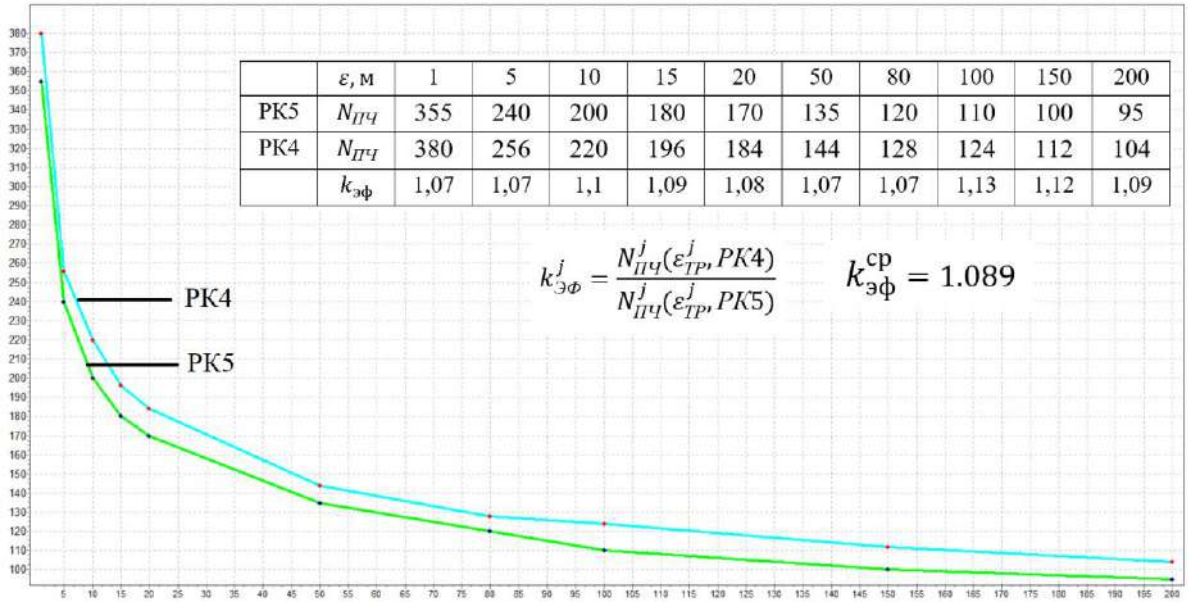
Рис. 6 Зависимость ошибки методов от колличества вычислений правых частей в диапазоне [0.5,20] м.



РК5	N	155	145	135	110	100	95
	ε, м	29.217	38.292	49.754	99.622	147.646	195.446
РК4	N	168	156	144	124	112	104
	ε, м	28.1	38.328	48.527	99.614	149.865	193.924

Рис. 7 Зависимость ошибки методов от колличества вычислений правых частей в диапазоне [20,200] м.

На основе данных, представленных на рисунках 6 и 7, для заданного набора требуемой точности решения задачи определения положения КЛА были получены данные для сравнительного анализа методов по критерию «заданная точность-минимальные вычислительные затраты», которые показаны на рисунке 8.



	ε, м	1	5	10	15	20	50	80	100	150	200
РК5	$N_{ПЧ}$	355	240	200	180	170	135	120	110	100	95
РК4	$N_{ПЧ}$	380	256	220	196	184	144	128	124	112	104
	$k_{эф}$	1,07	1,07	1,1	1,09	1,08	1,07	1,07	1,13	1,12	1,09

Рис. 8 Сравнительный анализ методов по критерию «Заданная точность-минимальные вычислительные затраты»

В первой строке таблице приведены значения требуемой точности решения задачи. Во 2-й и 3-й строках таблицы приведены необходимые для их достижения вычислительные затраты. Они показывают, что для всех значений требуемой точности расчета оптимальным является пяти-стадийный метод Рунге-Кутты-Мерсона 4-го порядка. В четвертой строке приведены значения коэффициента эффективности метода для каждого значения требуемой точности расчета. Они рассчитаны по формуле (3.2).

Среднее значения коэффициента эффективности метода Рунге-Кутты-Мерсона 4-го порядка по сравнению с классическим методом Рунге-Кутты 4-го порядка, который в настоящее время широко используется в практике баллистических расчетов, вычислено по формуле (6.1) и равно 1,09.

$$k_{ЭФ}^{СР} = \frac{1}{M}\sum_{j=1}^{M} k_{ЭФ}^{j} = k_{эф}^{ср} = \frac{1,07+1,07+1,1+1,09+1,08+1,07+1,07+1,13+1,12+1,09}{10} = 1.09 \quad (6.1)$$

где M – количество значений в наборе заданной точности решения задачи.

Это дает основание рекомендовать его для использования в алгоритмах бортовых и наземных системах управления полетом КЛА.

ЗАКЛЮЧЕНИЕ

При выполнении ВКР «Исследование алгоритмов и моделирование расчета параметров движения объекта» получены следующие результаты

1. На основе анализа действующих на КЛА сил построена математическая модель прогноза движения КЛА в гравитационном поле Земли.

2. Разработаны алгоритмы расчета параметров движения КЛА в общеземной системе координат.

3. Разработана методика сравнительной оценки эффективности алгоритмов численного интегрирования уравнений движения КЛА

4. Разработана программа, реализующая алгоритмы расчета параметров движения КЛА с использованием методов, Рунге-Кутты 2-го, 3-го и 4-го порядков, и обеспечивающая получение данных для их сравнительной оценки.

5. Для заданных начальных условий движения КЛА и заданных требований к точности решения задачи прогноза движения КЛА проведен сравнительный анализ алгоритмов, разработанных на основе выбранных методов. Определен оптимальный алгоритм, получена оценка его эффективности, на основании которой он может быть рекомендован для использования в вычислительных системах бортовых и наземных системах управления полетом КЛА.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Аппазов Р.Ф., Сытин О.Г. Методы проектирования траекторий носителей и спутников Земли. – М.: Наука, 1987 – 440 с.

2. Лебедев А.А., Герасюта Н.Ф. Баллистика ракет. – М.: Машиностроение, 1970 – 244 с.

3. «Параметры Земли 1990 года» (ПЗ-90.11). Справочный документ. – М.: ВТУ ГШ ВС РФ, 2014 – 52 с.

4. О единых государственных системах координат. Постановление Правительства РФ от 28 декабря 2012 г. № 1463. – М.: 2012.

5. Динамика летательных аппаратов в атмосфере. Термины, определения и обозначения. ГОСТ 20058-80. – М.: Издательство стандартов, 1981 – 51 с.

6. Корн Г., Корн Т. Справочник по математике для научных работников и инженеров. – М.: Наука, 1974 – 832 с.

7. Хайрер Э., Нерсетт С., Ваннер Г. Решение обыкновенных дифференциальных уравнений. Нежесткие задачи – М.: Мир, 1990 – 512 с.

8. Каханер Д. и др. Численные методы и программное обеспечение. – М.: Мир, 1998 – 575 с.

9. Макс Шлее, Qt 4.8. Профессиональное программирование на C++: [Электронный ресурс]. БХВ-Петербург, 2012. — 912 с. URL: http://www.cosmic-rays.ru/books61/Shlee_Qt_4_8.pdf.

10. СТО ЮУрГУ 04–2008 Стандарт организации. Курсовое и дипломное проектирование. Общие требования к содержанию и оформлению / составители: Т.И. Парубочая, Н.В. Сырейщикова, В.И. Гузеев, Л.В. Винокурова. – Челябинск: Изд-во ЮУрГУ, 2008. – 56 с.

ПРИЛОЖЕНИЕ А

Определения, обозначения и сокращения

В пояснительной записке применяют следующие термины с соответствующими определениями:

ВКР – выпускная квалификационная работа

ГПЗ – гравитационное поле Земли

ЛА – летательный аппарат

ОЗЭ – общеземной эллепсоид

КЛА – космический летательный аппарат

УД – уравнение движения

РК2 – метод Рунге-Кутты второго порядка

РК3 – метод Рунге-Кутты третьего порядка

РК4 – метод Рунге-Кутты четвертого порядка

РК5 – пятистадийный метод Рунге-Кутты-Мерсона четвертого порядка

N2 – колличество расчетов правых частей методом Рунге-Кутты второго порядка

N3 – колличество расчетов правых частей методом Рунге-Кутты третьего порядка

N4 – количество расчетов правых частей методом Рунге-Кутты четвертого порядка

N5 – количество расчетов правых частей методом Рунге-Кутты-Мерсона четвертого порядка

КЛА – космический летательный аппарат

ЛА – летательный аппарат

ПРИЛОЖЕНИЕ Б

Листинг ПО

Файл «calc.cpp», в котором находятся функции расчета ускорений гравитационного ускорений, переносного и кориолисова.

```
#include "calc.h"

calc::calc()
{

}

// нахождение гравитационного ускорения
double calc::gravx(double x, double y, double z) {
   double r = qSqrt(x*x+y*y+z*z);
   double mu = qPow(z/r,2);
   double aer = ae/r;
   double dq = -3*ge*qPow(aer,4)*J2;
   double q = -ge*qPow(aer,2)*(1-qPow(aer,2)*1.5*J2*(3*mu-1))-dq*mu;
   double gx = q*x/r;
   return gx;
}
double calc::gravy(double x, double y, double z) {
   double r = qSqrt(x*x+y*y+z*z);
   double mu = (z/r)*(z/r);
   double dq = -3*ge*qPow(ae/r,4)*J2;
   double q = -ge*qPow(ae/r,2)*(1-qPow(ae/r,2)*1.5*J2*(3*mu-1))-
dq*mu;
   double gy = q*y/r;
   return gy;
}
```

```
double calc::gravz(double x, double y, double z) {
   double r = qSqrt(x*x+y*y+z*z);
   double mu = (z/r)*(z/r);
   double dq = -3*ge*qPow(ae/r,4)*J2;
   double q = -ge*qPow(ae/r,2)*(1-qPow(ae/r,2)*1.5*J2*(3*mu-1))-
dq*mu;
   double gz = ((q+dq)*z)/r;
   return gz;
}

//-------------------------------------------------
// Вычисление кориолисово и переносного ускорения (НА ТРЕТЬЮ
ОСЬ ПРОЕКЦИЯ = 0)
double calc::korandperx(double x, double vy) {
   double wzx = omegaZ*(omegaZ*x+2*vy);
   return wzx;
}
double calc::korandpery(double y, double vx) {
   double wzy = omegaZ*(omegaZ*y-2*vx);
   return wzy;
}
```

ПРИЛОЖЕНИЕ В

Листинг ПО

Файл «mainwindow.cpp», в котором происходят все основные вычисления.

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}

MainWindow::~MainWindow()
{
    delete ui;
}



void MainWindow::on_result_released()
{
    calc* w = new calc; // объявляю локальную переменную для работы с классом calc
// РАБОТА С ФАЙЛАМИ
    QFile p2("C:\\1integration_result\\errorpoints2.txt");
    QFile p3("C:\\1integration_result\\errorpoints3.txt");
    QFile p4("C:\\1integration_result\\errorpoints4.txt");
    QFile p5("C:\\1integration_result\\errorpoints5.txt");
```

```
    QFile n2("C:\\1integration_result\\N2.txt");
    QFile n3("C:\\1integration_result\\N3.txt");
    QFile n4("C:\\1integration_result\\N4.txt");
    QFile n5("C:\\1integration_result\\N5.txt");
    p2.open(QIODevice::WriteOnly);
    p3.open(QIODevice::WriteOnly);
    p4.open(QIODevice::WriteOnly);
    p5.open(QIODevice::WriteOnly);
    n2.open(QIODevice::WriteOnly);
    n3.open(QIODevice::WriteOnly);
    n4.open(QIODevice::WriteOnly);
    n5.open(QIODevice::WriteOnly);
// ERROR RKM5 START----------------------------------------------------------------
    h = 1; // шаг интегрирования
    tn = ui->tn->text().toDouble(); // время начала интегрирования
    tk = ui->tk->text().toDouble(); // время конца интегрирования
    xn = ui->xn->text().toDouble(); // начальные координаты
    yn = ui->yn->text().toDouble();
    zn = ui->zn->text().toDouble();
    vxn = ui->vxn->text().toDouble(); // начальные скорости
    vyn = ui->vyn->text().toDouble();
    vzn = ui->vzn->text().toDouble();
    for (tn; tn<tk; tn=tn+h){
// Этап 1 ----------------------------------------------
    x = xn;       vx = vxn;
    y = yn;       vy = vyn;
    z = zn;       vz = vzn;
    gx = w->gravx(x, y, z); // нахождение проекций грав. ускорения на ось 1
    gy = w->gravy(x, y, z); // нахождение проекций грав. ускорения на ось 2
```

```
    gz = w->gravz(x, y, z); // нахождение проекций грав. ускорения на ось 3
    wzx = w->korandperx(x,vy); // проекция кориолисова и переносного ускорений на ось 1
    wzy = w->korandpery(y,vx); // проекция кориолисова и переносного ускорений на ось 2
    fq1vx = vxn;   fq1ax = gx+wzx;
    fq1vy = vyn;   fq1ay = gy+wzy;
    fq1vz = vzn;   fq1az = gz;
// проекция кориолисова и переносного ускорений на ось 3 wzz = 0
// Этап 2 ------------------------------------------------
    x = xn + h*fq1vx/3;
    y = yn + h*fq1vy/3;
    z = zn + h*fq1vz/3;
    vx = vxn + h*fq1ax/3;
    vy = vyn + h*fq1ay/3;
    vz = vzn + h*fq1az/3;
    gx = w->gravx(x, y, z); // нахождение проекций грав. ускорения на ось 1
    gy = w->gravy(x, y, z); // нахождение проекций грав. ускорения на ось 2
    gz = w->gravz(x, y, z); // нахождение проекций грав. ускорения на ось 3
    wzx = w->korandperx(x,vy); // проекция кориолисова и переносного ускорений на ось 1
    wzy = w->korandpery(y,vx); // проекция кориолисова и переносного ускорений на ось 2
    fq2vx = vx;   fq2ax = gx+wzx;
    fq2vy = vy;   fq2ay = gy+wzy;
    fq2vz = vz;   fq2az = gz;
// Этап 3 ------------------------------------------------
    x = xn + h*(fq1vx+fq2vx)/6;
    y = yn + h*(fq1vy+fq2vy)/6;
```

```
    z = zn + h*(fq1vz+fq2vz)/6;
    vx = vxn + h*(fq1ax+fq2ax)/6;
    vy = vyn + h*(fq1ay+fq2ay)/6;
    vz = vzn + h*(fq1az+fq2az)/6;
    gx = w->gravx(x, y, z); // нахождение проекций грав. ускорения на ось 1
    gy = w->gravy(x, y, z); // нахождение проекций грав. ускорения на ось 2
    gz = w->gravz(x, y, z); // нахождение проекций грав. ускорения на ось 3
    wzx = w->korandperx(x,vy); // проекция кориолисова и переносного ускорений на ось 1
    wzy = w->korandpery(y,vx); // проекция кориолисова и переносного ускорений на ось 2
    fq3vx = vx;   fq3ax = gx+wzx;
    fq3vy = vy;   fq3ay = gy+wzy;
    fq3vz = vz;   fq3az = gz;
// Этап 4 ------------------------------------------------
    x = xn + h*(fq1vx+3*fq3vx)/8;
    y = yn + h*(fq1vy+3*fq3vy)/8;
    z = zn + h*(fq1vz+3*fq3vz)/8;
    vx = vxn + h*(fq1ax+3*fq3ax)/8;
    vy = vyn + h*(fq1ay+3*fq3ay)/8;
    vz = vzn + h*(fq1az+3*fq3az)/8;
    gx = w->gravx(x, y, z); // нахождение проекций грав. ускорения на ось 1
    gy = w->gravy(x, y, z); // нахождение проекций грав. ускорения на ось 2
    gz = w->gravz(x, y, z); // нахождение проекций грав. ускорения на ось 3
    wzx = w->korandperx(x,vy); // проекция кориолисова и переносного ускорений на ось 1
    wzy = w->korandpery(y,vx); // проекция кориолисова и переносного ускорений на ось 2
    fq4vx = vx;   fq4ax = gx+wzx;
```

```
   fq4vy = vy;   fq4ay = gy+wzy;
    fq4vz = vz;   fq4az = gz;
// Этап 4.5 ------------------------------------------------
    fq45vx = xn + h*(fq1vx-3*fq3vx+4*fq4vx)/2; // =y(n+1)'
    fq45vy = yn + h*(fq1vy-3*fq3vy+4*fq4vy)/2;
    fq45vz = zn + h*(fq1vz-3*fq3vz+4*fq4vz)/2;
    fq45ax = vxn + h*(fq1ax-3*fq3ax+4*fq4ax)/2;
    fq45ay = vyn + h*(fq1ay-3*fq3ay+4*fq4ay)/2;
    fq45az = vzn + h*(fq1az-3*fq3az+4*fq4az)/2;

// Этап 5 ------------------------------------------------
    x = fq45vx;
    y = fq45vy;
    z = fq45vz;
    vx = fq45ax;
    vy = fq45ay;
    vz = fq45az;
    gx = w->gravx(x, y, z); // нахождение проекций грав. ускорения на ось 1
    gy = w->gravy(x, y, z); // нахождение проекций грав. ускорения на ось 2
    gz = w->gravz(x, y, z); // нахождение проекций грав. ускорения на ось 3
    wzx = w->korandperx(x,vy); // проекция кориолисова и переносного ускорений на ось 1
    wzy = w->korandpery(y,vx); // проекция кориолисова и переносного ускорений на ось 2
    fq5vx = vx;
    fq5vy = vy;
    fq5vz = vz;
    fq5ax = gx+wzx;
    fq5ay = gy+wzy;
```

```
    fq5az = gz;
    xn = xn + h*(fq1vx+4*fq4vx+fq5vx)/6;
    yn = yn + h*(fq1vy+4*fq4vy+fq5vy)/6;
    zn = zn + h*(fq1vz+4*fq4vz+fq5vz)/6;
    vxn = vxn + h*(fq1ax+4*fq4ax+fq5ax)/6;
    vyn = vyn + h*(fq1ay+4*fq4ay+fq5ay)/6;
    vzn = vzn + h*(fq1az+4*fq4az+fq5az)/6;
    N = N + 5;
    ex1 = xn;
    ey1 = yn;
    ez1 = zn;
    }
// ERROR RKM5 END-----------------------------------------------------------------


// RK2 START----------------------------------------------------------------------
    hk = ui->step_end->text().toDouble(); // начальный шаг интегрирования
    dh = ui->delta_step->text().toDouble();
    h = ui->step->text().toDouble(); // начальный шаг интегрирования
    remember_h = h; // переменная для запоминания h во внешнем цикле по h
    for(h; h<=hk; h = remember_h + dh){
    remember_h = h;
    tn = ui->tn->text().toDouble(); // время начала интегрирования
    tk = ui->tk->text().toDouble(); // время конца интегрирования
    xn = ui->xn->text().toDouble(); // начальные координаты
    yn = ui->yn->text().toDouble();
    zn = ui->zn->text().toDouble();
    vxn = ui->vxn->text().toDouble(); // начальные скорости
    vyn = ui->vyn->text().toDouble();
```

```
   vzn = ui->vzn->text().toDouble();
    N = 0; // сброс счетчика рассчета правых частей
    for (tn; tn<tk; tn=tn+h){
// ЭТАП 1 ----------------------
    if(tn+h>tk) h = h - (tn +h - tk);
    x = xn;       vx = vxn;
    y = yn;       vy = vyn;
    z = zn;       vz = vzn;
    gx = w->gravx(x, y, z); // нахождение проекций грав. ускорения на ось 1
    gy = w->gravy(x, y, z); // нахождение проекций грав. ускорения на ось 2
    gz = w->gravz(x, y, z); // нахождение проекций грав. ускорения на ось 3
    wzx = w->korandperx(x,vy); // проекция кориолисова и переносного ускорений на ось 1
    wzy = w->korandpery(y,vx); // проекция кориолисова и переносного ускорений на ось 2
    fq1vx = vxn;    fq1ax = gx + wzx;
    fq1vy = vyn;    fq1ay = gy +wzy;
    fq1vz = vzn;    fq1az = gz;
// ЭТАП 2 -------------------------
    x = xn + 0.5*h*fq1vx;
    y = yn + 0.5*h*fq1vy;
    z = zn + 0.5*h*fq1vz;
    vx = vxn + 0.5*h*fq1ax;
    vy = vyn + 0.5*h*fq1ay;
    vz = vzn + 0.5*h*fq1az;
    gx = w->gravx(x, y, z); // нахождение проекций грав. ускорения на ось 1
    gy = w->gravy(x, y, z); // нахождение проекций грав. ускорения на ось 2
    gz = w->gravz(x, y, z); // нахождение проекций грав. ускорения на ось 3
```

```
wzx = w->korandperx(x,vy); // проекция кориолисова и переносного ускорений на ось 1
wzy = w->korandpery(y,vx); // проекция кориолисова и переносного ускорений на ось 2
fq2vx = vx;          fq2ax = gx+wzx;
fq2vy = vy;          fq2ay = gy+wzy;
fq2vz = vz;          fq2az = gz;
xn = xn + h*fq2vx;   ex = xn;
yn = yn + h*fq2vy;   ey = yn;
zn = zn + h*fq2vz;   ez = zn;
vxn = vxn + fq2ax*h; evx = vxn;
vyn = vyn + fq2ay*h; evy = vyn;
vzn = vzn + fq2az*h; evz = vzn;
N = N + 2;
}
er_point2 = qSqrt(qPow(ex-ex1,2) + qPow(ey-ey1,2) + qPow(ez-ez1,2)); // расчет ошибки
p2.write(QString::number(er_point2,'f',3).toUtf8()+ "\n");          // записываю ошибку на текущем шаге в файл
n2.write(QString::number(N).toUtf8()+ "\n");
}
p2.close();
n2.close();
// RK2 END-----------------------------------------------------------------

// RK3 START---------------------------------------------------------------
    hk = ui->step_end->text().toDouble(); // начальный шаг интегрирования
    dh = ui->delta_step->text().toDouble();
    h = ui->step->text().toDouble(); // начальный шаг интегрирования
```

```
remember_h = ui->step->text().toDouble(); // переменная для запоминания h во внешнем цикле по h
for(h; h<=hk; h = remember_h + dh){
remember_h = h;
tn = ui->tn->text().toDouble(); // время начала интегрирования
tk = ui->tk->text().toDouble(); // время конца интегрирования
xn = ui->xn->text().toDouble(); // начальные координаты
yn = ui->yn->text().toDouble();
zn = ui->zn->text().toDouble();
vxn = ui->vxn->text().toDouble(); // начальные скорости
vyn = ui->vyn->text().toDouble();
vzn = ui->vzn->text().toDouble();
N = 0; // сброс счетчика рассчета правых частей
for (tn; tn<tk; tn=tn+h){
// ЭТАП 1 ----------------------
if(tn+h>tk) h = h - (tn +h - tk);
x = xn;       vx = vxn;
y = yn;       vy = vyn;
z = zn;       vz = vzn;
gx = w->gravx(x, y, z); // нахождение проекций грав. ускорения на ось 1
gy = w->gravy(x, y, z); // нахождение проекций грав. ускорения на ось 2
gz = w->gravz(x, y, z); // нахождение проекций грав. ускорения на ось 3
wzx = w->korandperx(x,vy); // проекция кориолисова и переносного ускорений на ось 1
wzy = w->korandpery(y,vx); // проекция кориолисова и переносного ускорений на ось 2
fq1vx = vxn;   fq1ax = gx+wzx;
fq1vy = vyn;   fq1ay = gy+wzy;
fq1vz = vzn;   fq1az = gz;
```

```
// Этап 2 ----------------------------------------------
      x = xn + 0.5*h*fq1vx;
      y = yn + 0.5*h*fq1vy;
      z = zn + 0.5*h*fq1vz;
      vx = vxn + 0.5*h*fq1ax;
      vy = vyn + 0.5*h*fq1ay;
      vz = vzn + 0.5*h*fq1az;
      gx = w->gravx(x, y, z); // нахождение проекций грав. ускорения на ось 1
      gy = w->gravy(x, y, z); // нахождение проекций грав. ускорения на ось 2
      gz = w->gravz(x, y, z); // нахождение проекций грав. ускорения на ось 3
      wzx = w->korandperx(x,vy); // проекция кориолисова и переносного ускорений на ось 1
      wzy = w->korandpery(y,vx); // проекция кориолисова и переносного ускорений на ось 2
      fq2vx = vx;   fq2ax = gx+wzx;
      fq2vy = vy;   fq2ay = gy+wzy;
      fq2vz = vz;   fq2az = gz;
   // Этап 3 ----------------------------------------------
      x = xn + h*(2*fq2vx-fq1vx);
      y = yn + h*(2*fq2vy-fq1vy);
      z = zn + h*(2*fq2vz-fq1vz);
      vx = vxn + h*(2*fq2ax-fq1ax);
      vy = vyn + h*(2*fq2ay-fq1ay);
      vz = vzn + h*(2*fq2az-fq1az);
      gx = w->gravx(x, y, z); // нахождение проекций грав. ускорения на ось 1
      gy = w->gravy(x, y, z); // нахождение проекций грав. ускорения на ось 2
      gz = w->gravz(x, y, z); // нахождение проекций грав. ускорения на ось 3
      wzx = w->korandperx(x,vy); // проекция кориолисова и переносного ускорений на ось 1
```

```
    wzy = w->korandpery(y,vx); // проекция кориолисова и переносного ускорений на ось 2
        fq3vx = vx;   fq3ax = gx+wzx;
        fq3vy = vy;   fq3ay = gy+wzy;
        fq3vz = vz;   fq3az = gz;
        xn = xn + h*(fq1vx+4*fq2vx+fq3vx)/6;
        yn = yn + h*(fq1vy+4*fq2vy+fq3vy)/6;
        zn = zn + h*(fq1vz+4*fq2vz+fq3vz)/6;
        vxn = vxn + h*(fq1ax+4*fq2ax+fq3ax)/6;
        vyn = vyn + h*(fq1ay+4*fq2ay+fq3ay)/6;
        vzn = vzn + h*(fq1az+4*fq2az+fq3az)/6;
        N = N + 3;
        ex = xn;   evx = vxn;
        ey = yn;   evy = vyn;
        ez = zn;   evz = vzn;
        }
        er_point3 = qSqrt(qPow(ex-ex1,2) + qPow(ey-ey1,2) + qPow(ez-ez1,2));
        p3.write(QString::number(er_point3,'f',3).toUtf8()+  "\n");        // записываю ошибку на текущем шаге в файл
        n3.write(QString::number(N).toUtf8()+ "\n");
        }
        p3.close();
        n3.close();
// RK3 END-----------------------------------------------------------------

// RK4 START---------------------------------------------------------------
    hk = ui->step_end->text().toDouble(); // начальный шаг интегрирования
    dh = ui->delta_step->text().toDouble();
    h = ui->step->text().toDouble(); // начальный шаг интегрирования
```

```
remember_h = ui->step->text().toDouble(); // переменная для запоминания h во внешнем цикле по h
for(h; h<=hk; h = remember_h + dh){
remember_h = h;
tn = ui->tn->text().toDouble(); // время начала интегрирования
tk = ui->tk->text().toDouble(); // время конца интегрирования
xn = ui->xn->text().toDouble(); // начальные координаты
yn = ui->yn->text().toDouble();
zn = ui->zn->text().toDouble();
vxn = ui->vxn->text().toDouble(); // начальные скорости
vyn = ui->vyn->text().toDouble();
vzn = ui->vzn->text().toDouble();
N = 0; // сброс счетчика рассчета правых частей
for (tn; tn<tk; tn=tn+h){
// ЭТАП 1 ----------------------
 if(tn+h>tk) h = h - (tn +h - tk);
 x = xn;       vx = vxn;
 y = yn;       vy = vyn;
 z = zn;       vz = vzn;
 gx = w->gravx(x, y, z); // нахождение проекций грав. ускорения на ось 1
 gy = w->gravy(x, y, z); // нахождение проекций грав. ускорения на ось 2
 gz = w->gravz(x, y, z); // нахождение проекций грав. ускорения на ось 3
 wzx = w->korandperx(x,vy); // проекция кориолисова и переносного ускорений на ось 1
 wzy = w->korandpery(y,vx); // проекция кориолисова и переносного ускорений на ось 2
 fq1vx = vxn;   fq1ax = gx+wzx;
 fq1vy = vyn;   fq1ay = gy+wzy;
 fq1vz = vzn;   fq1az = gz;
```

```
// Этап 2 ---------------------------------------------
   x = xn + 0.5*h*fq1vx;
   y = yn + 0.5*h*fq1vy;
   z = zn + 0.5*h*fq1vz;
   vx = vxn + 0.5*h*fq1ax;
   vy = vyn + 0.5*h*fq1ay;
   vz = vzn + 0.5*h*fq1az;
   gx = w->gravx(x, y, z); // нахождение проекций грав. ускорения на ось 1
   gy = w->gravy(x, y, z); // нахождение проекций грав. ускорения на ось 2
   gz = w->gravz(x, y, z); // нахождение проекций грав. ускорения на ось 3
   wzx = w->korandperx(x,vy); // проекция кориолисова и переносного ускорений на ось 1
   wzy = w->korandpery(y,vx); // проекция кориолисова и переносного ускорений на ось 2
   fq2vx = vx;   fq2ax = gx+wzx;
   fq2vy = vy;   fq2ay = gy+wzy;
   fq2vz = vz;   fq2az = gz;
// Этап 3 ---------------------------------------------
   x = xn + 0.5*h*fq2vx;
   y = yn + 0.5*h*fq2vy;
   z = zn + 0.5*h*fq2vz;
   vx = vxn + 0.5*h*fq2ax;
   vy = vyn + 0.5*h*fq2ay;
   vz = vzn + 0.5*h*fq2az;
   gx = w->gravx(x, y, z); // нахождение проекций грав. ускорения на ось 1
   gy = w->gravy(x, y, z); // нахождение проекций грав. ускорения на ось 2
   gz = w->gravz(x, y, z); // нахождение проекций грав. ускорения на ось 3
   wzx = w->korandperx(x,vy); // проекция кориолисова и переносного ускорений на ось 1
```

```
 wzy = w->korandpery(y,vx); // проекция кориолисова и переносного ускорений на ось 2
    fq3vx = vx;   fq3ax = gx+wzx;
    fq3vy = vy;   fq3ay = gy+wzy;
    fq3vz = vz;   fq3az = gz;
// Этап 4 ------------------------------------------------
    x = xn + h*fq3vx;
    y = yn + h*fq3vy;
    z = zn + h*fq3vz;
    vx = vxn + h*fq3ax;
    vy = vyn + h*fq3ay;
    vz = vzn + h*fq3az;
    gx = w->gravx(x, y, z); // нахождение проекций грав. ускорения на ось 1
    gy = w->gravy(x, y, z); // нахождение проекций грав. ускорения на ось 2
    gz = w->gravz(x, y, z); // нахождение проекций грав. ускорения на ось 3
    wzx = w->korandperx(x,vy); // проекция кориолисова и переносного ускорений на ось 1
    wzy = w->korandpery(y,vx); // проекция кориолисова и переносного ускорений на ось 2
    fq4vx = vx;   fq4ax = gx+wzx;
    fq4vy = vy;   fq4ay = gy+wzy;
    fq4vz = vz;   fq4az = gz;
    xn = xn + h*(fq1vx+2*(fq2vx+fq3vx)+fq4vx)/6;
    yn = yn + h*(fq1vy+2*(fq2vy+fq3vy)+fq4vy)/6;
    zn = zn + h*(fq1vz+2*(fq2vz+fq3vz)+fq4vz)/6;
    vxn = vxn + h*(fq1ax+2*(fq2ax+fq3ax)+fq4ax)/6;
    vyn = vyn + h*(fq1ay+2*(fq2ay+fq3ay)+fq4ay)/6;
    vzn = vzn + h*(fq1az+2*(fq2az+fq3az)+fq4az)/6;
    N = N + 4;
```

```
    ex = xn;   evx = vxn;
    ey = yn;   evy = vyn;
    ez = zn;   evz = vzn;
    }
    er_point4 = qSqrt(qPow(ex-ex1,2) + qPow(ey-ey1,2) + qPow(ez-ez1,2));
    p4.write(QString::number(er_point4,'f',3).toUtf8()+   "\n");          //  записываю
ошибку на текущем шаге в файл
    n4.write(QString::number(N).toUtf8()+ "\n");
    }
    p4.close();
    n4.close();
// RK4 END-----------------------------------------------------------------
// RKM5 START--------------------------------------------------------------
        hk = ui->step_end->text().toDouble(); // начальный шаг интегрирования
        dh = ui->delta_step->text().toDouble();
        h = ui->step->text().toDouble(); // начальный шаг интегрирования
        remember_h = ui->step->text().toDouble(); // переменная для запоминания h
во внешнем цикле по h
        for(h; h<=hk; h = remember_h + dh){
        remember_h = h;
        tn = ui->tn->text().toDouble(); // время начала интегрирования
        tk = ui->tk->text().toDouble(); // время конца интегрирования
        xn = ui->xn->text().toDouble(); // начальные координаты
        yn = ui->yn->text().toDouble();
        zn = ui->zn->text().toDouble();
        vxn = ui->vxn->text().toDouble(); // начальные скорости
        vyn = ui->vyn->text().toDouble();
        vzn = ui->vzn->text().toDouble();
        N = 0; // сброс счетчика рассчета правых частей
```

```
 for (tn; tn<tk; tn=tn+h){
// Этап 1 ------------------------------------------------
   if(tn+h>tk) h = h - (tn +h - tk);
   x = xn;       vx = vxn;
   y = yn;       vy = vyn;
   z = zn;       vz = vzn;
   gx = w->gravx(x, y, z); // нахождение проекций грав. ускорения на ось 1
   gy = w->gravy(x, y, z); // нахождение проекций грав. ускорения на ось 2
   gz = w->gravz(x, y, z); // нахождение проекций грав. ускорения на ось 3
   wzx = w->korandperx(x,vy); // проекция кориолисова и переносного ускорений на ось 1
   wzy = w->korandpery(y,vx); // проекция кориолисова и переносного ускорений на ось 2
   fq1vx = vxn;   fq1ax = gx+wzx;
   fq1vy = vyn;   fq1ay = gy+wzy;
   fq1vz = vzn;   fq1az = gz;
// проекция кориолисова и переносного ускорений на ось 3 wzz = 0
// Этап 2 ------------------------------------------------
   x = xn + h*fq1vx/3;
   y = yn + h*fq1vy/3;
   z = zn + h*fq1vz/3;
   vx = vxn + h*fq1ax/3;
   vy = vyn + h*fq1ay/3;
   vz = vzn + h*fq1az/3;
   gx = w->gravx(x, y, z); // нахождение проекций грав. ускорения на ось 1
   gy = w->gravy(x, y, z); // нахождение проекций грав. ускорения на ось 2
   gz = w->gravz(x, y, z); // нахождение проекций грав. ускорения на ось 3
   wzx = w->korandperx(x,vy); // проекция кориолисова и переносного ускорений на ось 1
```

```
        wzy = w->korandpery(y,vx); // проекция кориолисова и переносного ускорений на ось 2
        fq2vx = vx;   fq2ax = gx+wzx;
        fq2vy = vy;   fq2ay = gy+wzy;
        fq2vz = vz;   fq2az = gz;
    // Этап 3 ------------------------------------------------
        x = xn + h*(fq1vx+fq2vx)/6;
        y = yn + h*(fq1vy+fq2vy)/6;
        z = zn + h*(fq1vz+fq2vz)/6;
        vx = vxn + h*(fq1ax+fq2ax)/6;
        vy = vyn + h*(fq1ay+fq2ay)/6;
        vz = vzn + h*(fq1az+fq2az)/6;
        gx = w->gravx(x, y, z); // нахождение проекций грав. ускорения на ось 1
        gy = w->gravy(x, y, z); // нахождение проекций грав. ускорения на ось 2
        gz = w->gravz(x, y, z); // нахождение проекций грав. ускорения на ось 3
        wzx = w->korandperx(x,vy); // проекция кориолисова и переносного ускорений на ось 1
        wzy = w->korandpery(y,vx); // проекция кориолисова и переносного ускорений на ось 2
        fq3vx = vx;   fq3ax = gx+wzx;
        fq3vy = vy;   fq3ay = gy+wzy;
        fq3vz = vz;   fq3az = gz;
    // Этап 4 ------------------------------------------------
        x = xn + h*(fq1vx+3*fq3vx)/8;
        y = yn + h*(fq1vy+3*fq3vy)/8;
        z = zn + h*(fq1vz+3*fq3vz)/8;
        vx = vxn + h*(fq1ax+3*fq3ax)/8;
        vy = vyn + h*(fq1ay+3*fq3ay)/8;
        vz = vzn + h*(fq1az+3*fq3az)/8;
```

```
gx = w->gravx(x, y, z); // нахождение проекций грав. ускорения на ось 1
gy = w->gravy(x, y, z); // нахождение проекций грав. ускорения на ось 2
gz = w->gravz(x, y, z); // нахождение проекций грав. ускорения на ось 3
wzx = w->korandperx(x,vy); // проекция кориолисова и переносного ускорений на ось 1
wzy = w->korandpery(y,vx); // проекция кориолисова и переносного ускорений на ось 2
fq4vx = vx;   fq4ax = gx+wzx;
fq4vy = vy;   fq4ay = gy+wzy;
fq4vz = vz;   fq4az = gz;
// Этап 4.5 ------------------------------------------------
fq45vx = xn + h*(fq1vx-3*fq3vx+4*fq4vx)/2; // =y(n+1)'
fq45vy = yn + h*(fq1vy-3*fq3vy+4*fq4vy)/2;
fq45vz = zn + h*(fq1vz-3*fq3vz+4*fq4vz)/2;
fq45ax = vxn + h*(fq1ax-3*fq3ax+4*fq4ax)/2;
fq45ay = vyn + h*(fq1ay-3*fq3ay+4*fq4ay)/2;
fq45az = vzn + h*(fq1az-3*fq3az+4*fq4az)/2;

// Этап 5 ------------------------------------------------
x = fq45vx;
y = fq45vy;
z = fq45vz;
vx = fq45ax;
vy = fq45ay;
vz = fq45az;
gx = w->gravx(x, y, z); // нахождение проекций грав. ускорения на ось 1
gy = w->gravy(x, y, z); // нахождение проекций грав. ускорения на ось 2
gz = w->gravz(x, y, z); // нахождение проекций грав. ускорения на ось 3
```

```
wzx = w->korandperx(x,vy); // проекция кориолисова и переносного ускорений на ось 1
wzy = w->korandpery(y,vx); // проекция кориолисова и переносного ускорений на ось 2
fq5vx = vx;   fq5ax = gx+wzx;
fq5vy = vy;   fq5ay = gy+wzy;
fq5vz = vz;   fq5az = gz;
xn = xn + h*(fq1vx+4*fq4vx+fq5vx)/6;
yn = yn + h*(fq1vy+4*fq4vy+fq5vy)/6;
zn = zn + h*(fq1vz+4*fq4vz+fq5vz)/6;
vxn = vxn + h*(fq1ax+4*fq4ax+fq5ax)/6;
vyn = vyn + h*(fq1ay+4*fq4ay+fq5ay)/6;
vzn = vzn + h*(fq1az+4*fq4az+fq5az)/6;
N = N + 5;
ex = xn;   evx = vxn;
ey = yn;   evy = vyn;
ez = zn;   evz = vzn;
}
er_point5 = qSqrt(qPow(ex-ex1,2) + qPow(ey-ey1,2) + qPow(ez-ez1,2));
p5.write(QString::number(er_point5,'f',3).toUtf8()+  "\n");        // записываю ошибку на текущем шаге в файл
n5.write(QString::number(N).toUtf8()+ "\n");
}
p5.close();
n5.close();
// RKM5 END----------------------------------------------------------------
}
```

ПРИЛОЖЕНИЕ Г

Полученые результаты ошибки и колличество расчетов правых частей.

	РК2		РК3		РК4		РК5	
h,c	$\varepsilon, м$	$N_{ПЧ}$	$\varepsilon, м$	$N_{ПЧ}$	$\varepsilon, м$	$N_{ПЧ}$	$\varepsilon, м$	$N_{ПЧ}$
1	5.218	5102	0.003	7653	0.000	10204	0.000	12755
2	20.848	2552	0.024	3828	0.000	5104	0.000	6380
3	46.865	1702	0.079	2553	0.000	3404	0.000	4255
4	83.227	1276	0.187	1914	0.000	2552	0.000	3190
5	129.955	1022	0.365	1533	0.001	2044	0.000	2555
6	186.974	852	0.629	1278	0.002	1704	0.001	2130
7	254.090	730	0.997	1095	0.004	1460	0.001	1825
8	331.723	638	1.485	957	0.008	1276	0.002	1595
9	419.149	568	2.108	852	0.012	1136	0.004	1420
10	517.588	512	2.888	768	0.019	1024	0.006	1280
11	625.517	464	3.834	696	0.027	928	0.008	1160
12	742.718	426	4.961	639	0.039	852	0.012	1065
13	871.653	394	6.300	591	0.053	788	0.016	985
14	1010.018	366	7.851	549	0.071	732	0.022	915
15	1159.503	342	9.643	513	0.094	684	0.029	855
16	1315.111	320	11.653	480	0.122	640	0.038	800
17	1486.692	302	13.976	453	0.155	604	0.048	755
18	1661.110	284	16.509	426	0.196	568	0.060	710
19	1850.494	270	19.396	405	0.243	540	0.075	675
20	2045.658	256	22.537	384	0.299	512	0.092	640
21	2253.249	244	26.033	366	0.364	488	0.112	610
22	2477.775	232	29.936	348	0.437	464	0.134	580
23	2703.642	222	34.101	333	0.523	444	0.161	555
24	2936.942	214	38.629	321	0.621	428	0.190	535
25	3192.196	206	43.660	309	0.730	412	0.224	515
26	3446.279	198	48.963	297	0.855	396	0.262	495
27	3700.783	190	54.531	285	0.998	380	0.306	475
28	3989.575	184	60.876	276	1.151	368	0.352	460
29	4278.083	176	67.489	264	1.325	352	0.405	440

30	4575.791	172	74.591	258	1.517	344	0.463	430
31	4864.637	166	81.866	249	1.735	332	0.530	415
32	5172.366	160	89.657	240	1.976	320	0.603	400
33	5500.257	156	98.264	234	2.231	312	0.680	390
34	5855.593	152	107.585	228	2.507	304	0.763	380
35	6183.464	146	116.724	219	2.826	292	0.861	365
36	6531.784	142	126.638	213	3.167	284	0.964	355
37	6906.904	138	137.466	207	3.527	276	1.072	345
38	7273.536	136	148.567	204	3.924	272	1.192	340
39	7624.544	132	159.624	198	4.372	264	1.327	330
40	8017.764	128	171.740	192	4.845	256	1.470	320
41	8428.959	126	185.005	189	5.332	252	1.616	315
42	8815.955	122	197.719	183	5.897	244	1.788	305
43	9237.020	120	212.030	180	6.467	240	1.958	300
44	9710.539	116	227.595	174	7.067	232	2.137	290
45	10081.568	114	241.227	171	7.787	228	2.355	285
46	10521.480	112	257.185	168	8.497	224	2.568	280
47	10995.860	110	274.306	165	9.243	220	2.790	275
48	11484.798	108	292.138	162	10.038	216	3.026	270
49	11978.173	106	310.521	159	10.890	212	3.280	265
50	12471.475	104	329.390	156	11.803	208	3.552	260
51	12962.636	102	348.701	153	12.780	204	3.843	255
52	13450.305	100	368.413	150	13.828	200	4.156	250
53	13933.869	98	388.497	147	14.949	196	4.490	245
54	14415.198	96	408.948	144	16.148	192	4.849	240
55	14901.833	94	429.867	141	17.426	188	5.231	235
56	15411.133	92	451.660	138	18.775	184	5.635	230
57	15974.823	90	475.450	135	20.157	180	6.046	225
58	16643.556	88	503.791	132	21.448	176	6.418	220
59	17116.135	88	526.580	132	23.059	176	6.899	220
60	17612.034	86	549.755	129	24.781	172	7.416	215
61	18241.554	84	577.431	126	26.453	168	7.911	210
62	18881.723	84	607.629	126	28.100	168	8.386	210
63	19352.745	82	631.434	123	30.154	164	9.005	205

64	20038.010	80	662.469	120	32.056	160	9.564	200
65	20633.356	80	692.968	120	34.056	160	10.145	200
66	21169.117	78	719.601	117	36.404	156	10.853	195
67	21976.728	78	758.348	117	38.328	156	11.394	195
68	22413.082	76	783.057	114	41.025	152	12.210	190
69	23276.709	74	824.097	111	43.150	148	12.811	185
70	23709.325	74	850.330	111	46.063	148	13.686	185
71	24543.270	72	890.808	108	48.527	144	14.396	180
72	25028.745	72	920.411	108	51.588	144	15.305	180
73	25909.417	70	964.022	105	54.216	140	16.057	175
74	26364.413	70	993.022	105	57.644	140	17.082	175
75	27360.714	70	1043.658	105	60.232	140	17.796	175
76	27731.613	68	1068.642	102	64.247	136	19.020	170
77	28666.36	68	1119.194	102	67.162	136	19.824	170
78	29177.02	66	1149.319	99	71.334	132	21.093	165
79	29966.46	66	1196.232	99	74.753	132	22.052	165
80	30794.47	64	1240.636	96	78.610	128	23.184	160
81	31321.90	64	1276.408	96	82.984	128	24.475	160
82	32353.23	64	1334.343	96	86.457	128	25.425	160
83	32864.03	62	1365.972	93	91.585	124	26.987	155
84	33623.18	62	1414.946	93	95.867	124	28.195	155
85	34773.75	62	1479.360	93	99.614	124	29.217	155
86	35111.57	60	1504.779	90	105.734	120	31.095	150
87	35977.74	60	1560.614	90	110.301	120	32.361	150
88	37140.01	58	1627.266	87	114.585	116	33.541	145
89	37469.08	58	1653.277	87	121.393	116	35.626	145
90	38344.35	58	1711.685	87	126.496	116	37.040	145
91	39564.72	58	1784.228	87	131.119	116	38.292	145
92	39942.89	56	1812.004	84	138.646	112	40.604	140
93	40728.02	56	1867.917	84	144.604	112	42.279	140
94	41902.17	56	1941.849	84	149.865	112	43.696	140
95	42621.13	54	1987.136	81	157.208	108	45.900	135
96	43195.78	54	2031.923	81	164.641	108	48.078	135
97	44222.25	54	2102.588	81	170.814	108	49.754	135

98	45535.83	54	2185.477	81	176.588	108	51.305	135
99	45916.75	52	2213.965	78	186.111	104	54.234	130
100	46647.50	52	2270.609	78	193.924	104	56.456	130
101	47776.76	52	2349.237	78	200.746	104	58.288	130
102	49153.00	52	2438.663	78	207.222	104	60.022	130
103	49462.27	50	2463.425	75	218.270	100	63.431	125
104	50200.27	50	2522.492	75	227.141	100	65.953	125
105	51325.97	50	2604.411	75	234.933	100	68.042	125
106	52712.45	50	2698.389	75	242.323	100	70.007	125
107	53280.37	48	2738.121	72	253.781	96	73.492	120
108	53880.76	48	2788.819	72	264.557	96	76.640	120
109	54886.31	48	2867.484	72	273.790	96	79.150	120
110	56200.01	48	2962.536	72	282.383	96	81.420	120
111	57566.64	46	3057.247	69	291.374	92	83.882	115
112	57867.01	46	3082.224	69	305.632	92	88.267	115
113	58600.94	46	3144.891	69	317.331	92	91.600	115
114	59699.84	46	3232.562	69	327.709	92	94.385	115
115	61077.93	46	3335.416	69	337.510	92	96.959	115
116	62555.11	44	3441.869	66	347.419	88	99.622	110
117	62830.31	44	3463.710	66	363.967	88	104.726	110
118	63518.27	44	3524.400	66	377.744	88	108.689	110
119	64563.98	44	3612.267	66	389.956	88	111.988	110
120	65899.77	44	3717.857	66	401.411	88	114.989	110
121	67440.32	44	3834.409	66	412.495	88	117.885	110
122	68296.40	42	3896.513	63	428.213	84	122.546	105
123	68757.94	42	3936.962	63	445.771	84	127.810	105
124	69587.77	42	4011.427	63	460.995	84	132.077	105
125	70737.83	42	4110.164	63	474.886	84	135.775	105
126	72149.14	42	4225.255	63	488.103	84	139.211	105
127	73747.92	42	4351.029	63	500.953	84	142.545	105
128	74740.12	40	4425.300	60	518.401	80	147.646	100
129	75158.86	40	4460.734	60	539.219	80	153.927	100
130	75927.29	40	4531.702	60	557.442	80	159.097	100
131	77008.26	40	4629.299	60	574.052	80	163.560	100

132	78356.88	40	4745.991	60	589.761	80	167.646	100
133	79917.95	40	4875.858	60	604.987	80	171.563	100
134	81622.75	40	5015.011	60	619.869	80	175.408	100
135	82236.49	38	5058.776	57	643.302	76	182.479	95
136	82739.50	38	5102.737	57	666.918	76	189.504	95
137	83561.38	38	5180.376	57	688.060	76	195.446	95
138	84672.01	38	5284.044	57	707.596	76	200.671	95
139	86035.32	38	5407.153	57	726.178	76	205.484	95
140	87607.42	38	5544.350	57	744.218	76	210.092	95
141	89334.41	38	5691.806	57	761.890	76	214.606	95
142	90679.42	36	5801.982	54	783.311	72	220.602	90
143	91022.20	36	5824.237	54	812.938	72	229.616	90
144	91670.89	36	5884.125	54	839.508	72	237.321	90
145	92604.33	36	5974.525	54	863.889	72	244.038	90
146	93797.62	36	6089.085	54	886.811	72	250.091	90
147	95221.14	36	6222.288	54	908.828	72	255.744	90
148	96839.17	36	6369.577	54	930.296	72	261.183	90
149	98608.33	36	6527.572	54	951.379	72	266.514	90
150	100475.65	36	6694.367	54	972.062	72	271.783	90
151	100824.03	34	6709.182	51	1007.485	68	282.592	85
152	101351.94	34	6752.633	51	1040.526	68	292.362	85
153	102149.78	34	6829.427	51	1070.934	68	300.945	85
154	103200.19	34	6933.774	51	1099.429	68	308.636	85
155	104482.83	34	7060.473	51	1126.613	68	315.715	85
156	105973.59	34	7204.966	51	1152.948	68	322.410	85
157	107643.73	34	7363.439	51	1178.739	68	328.885	85
158	109458.68	34	7532.978	51	1204.129	68	335.236	85
159	111376.77	34	7711.777	51	1229.115	68	341.514	85
160	112354.89	32	7788.116	48	1264.677	64	351.856	80
161	112769.46	32	7811.365	48	1305.903	64	364.228	80
162	113434.88	32	7870.185	48	1344.096	64	375.251	80
163	114339.31	32	7959.419	48	1379.928	64	385.166	80
164	115469.05	32	8074.339	48	1414.002	64	394.235	80
165	116808.14	32	8210.662	48	1446.827	64	402.701	80

166	118337.88	32	8364.587	48	1478.803	64	410.766	80
167	120036.12	32	8532.861	48	1510.207	64	418.581	80
168	121876.60	32	8712.886	48	1541.188	64	426.246	80
169	123828.02	32	8902.858	48	1571.769	64	433.816	80
170	125853.10	32	9101.950	48	1601.863	64	441.318	80
171	126244.74	30	9108.342	45	1652.547	60	456.625	75
172	126740.32	30	9137.311	45	1701.382	60	471.073	75
173	127458.38	30	9199.785	45	1747.341	60	484.195	75
174	128390.21	30	9291.522	45	1790.993	60	496.200	75
175	129525.77	30	9408.596	45	1832.857	60	507.314	75
176	130853.51	30	9547.403	45	1873.389	60	517.757	75
177	132359.98	30	9704.682	45	1912.963	60	527.719	75
178	134029.50	30	9877.551	45	1951.863	60	537.352	75
179	135843.66	30	10063.565	45	1990.274	60	546.771	75
180	137780.83	30	10260.801	45	2028.282	60	556.050	75
181	139815.57	30	10467.959	45	2065.872	60	565.231	75
182	141917.92	30	10684.504	45	2102.936	60	574.336	75
183	142590.12	28	10716.264	42	2160.856	56	591.427	70
184	143046.85	28	10729.187	42	2221.484	56	609.395	70
185	143702.27	28	10775.705	42	2279.113	56	625.975	70
186	144550.62	28	10852.247	42	2334.231	56	641.322	70
187	145585.33	28	10955.468	42	2387.300	56	655.622	70
188	146798.93	28	11082.253	42	2438.743	56	669.072	70
189	148182.80	28	11229.719	42	2488.935	56	681.856	70
190	149727.04	28	11395.231	42	2538.194	56	694.139	70
191	151420.18	28	11576.423	42	2586.767	56	706.059	70
192	153248.95	28	11771.235	42	2634.829	56	717.723	70
193	155197.89	28	11977.963	42	2682.479	56	729.210	70
194	157249.07	28	12195.321	42	2729.733	56	740.569	70
195	159381.62	28	12422.524	42	2776.529	56	751.828	70
196	161571.31	28	12659.392	42	2822.734	56	763.003	70
197	162310.12	26	12689.579	39	2893.585	52	783.669	65
198	162763.20	26	12687.380	39	2969.004	52	805.915	65
199	163390.00	26	12717.900	39	3041.445	52	826.761	65
200	164186.85	26	12778.215	39	3111.309	52	846.312	65

НЕ УДАЛЯТЬ