МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

"САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА"

(САМАРСКИЙ УНИВЕРСИТЕТ)

Институт ракетно–космической техники

Кафедра космического машиностроения

# ПОЯСНТИТЕЛЬНАЯ ЗАПИСКА

к лабораторным работам по дисциплине

«Основы проектирования космических аппаратов с электроракетными двигателями»

Выполнил студент группы 1408

Хайруллин И.И.

Преподаватели

д.т.н., профессор Салмин В.В.

к.т.н., доцент Четвериков А.С.

Самара 2022

# РЕФЕРАТ

Пояснительная записка 69 стр., 17 рисунков, 2 таблицы 2 приложения, 4 источника.

МЕЖОРБИТАЛЬНЫЙ ТРАНСПОРТНЫЙ АППАРАТ, ЭЛЕКТРОРЕАКТИВНЫЙ ДВИГАТЕЛЬ, ДВИГАТЕЛЬНАЯ УСТАНОВКА, НАКЛОНЕНИЕ, УГОЛ ОРИЕНТАЦИИ ТЯГИ, СКОРОСТЬ ИСТЕЧЕНИЯ, ТЯГА, НАЧАЛЬНОЕ УСКОРЕНИЕ, СОЛНЕЧНЫЕ БАТАРЕИ.

Целью работы является создание программы, выполняющей расчет и моделирование межорбитального транспортного аппарата (МТА) с электрореактивными двигателями (ЭРД) с заданными начальными условиями.

Объектом исследования является МТА с ЭРД. Проведён проектно–баллистический расчёт перелётов межорбитального транспортного аппарата с солнечной энергоустановкой с низкой околоземной орбиты на геостационарную орбиту. Расчёты проводились для МТА с использованием разных двигателей. Выбраны основные проектные параметры, сформирован и отображен проектный облик МТА с использованием твердотельного моделирования в пакете Solid Works.

СОДЕРЖАНИЕ

# ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

РН – ракета–носитель;

БВ – блок выведения;

ПН – полезная нагрузка;

КБ – конструкторское бюро;

ГСО – геостационарная орбита;

СПД – стационарный плазменный двигатель;

ЭРДУ – электроракетная двигательная установка;

ДБ – двигательные блоки;

СХП – система хранения и подачи;

МТА – межорбитальный транспортный аппарат;

КА – космический аппарат;

ДУ – двигательная установка;

ЭУ – энергетическая установка;

СЭРДУ – солнечная электрореактивная двигательная установка;

СБ – солнечная батарея;

РН – ракета носитель;

СЭУ – солнечная энергетическая установка.

# ВВЕДЕНИЕ

В последние годы повысился интерес к созданию в космосе крупногабаритных конструкций, в первую очередь космических станций, спутниковых систем наблюдения и др., размещенных на высоких орбитах, в том числе на геостационарной. Реализуемость этих проектов в значительной степени зависит от эффективности МТА, доставляющих элементы конструкции с низкой околоземной орбиты на орбиту функционирования.

Одним из возможных путей решения этой задачи является использование для космических миссий перспективных двигательных систем с высокими техническими данными, основанных на новых физических принципах. К таким системам относятся ЭРД, работающие на принципе ускорения рабочего тела в электростатических или электромагнитных полях. Эти двигатели создают реактивное ускорение существенно меньше гравитационного ускорения на поверхности Земли, поэтому их, традиционно, называют двигателями малой тяги.

МТА с солнечными ЭРДУ рассматриваются как основное транспортное средство для самых различных целей – от транспортировки полезной нагрузки (ПН) на высокие околоземные орбиты до перелетов к другим телам Солнечной системы. Наиболее целесообразно применение ЭРД для многоразовых МТА, совершающих перелеты между низкой и высокими орбитами. Эффективность их использования увеличивается с увеличением числа повторных рейсов «орбита–орбита» [2].

Целью работы является создание программы, выполняющей расчет и моделирование МТА с ЭРД с заданными начальными условиями.

Задачами данной работы являются изучение методики расчета оптимальных баллистических схем полета и проектных характеристик околоземных МТА с ЭРД малой тяги, формирование и отображение проектного облика МТА с использованием системы твердотельного моделирования Solid Works.

# 1 Проектный облик МТА с ЭРДУ

Исследования, выполненные различными авторами, позволяют указать класс задач, для решения которых электрореактивные двигатели имеют преимущество по сравнению с термохимическими или ядерными. К этим задачам, в том числе, относятся межорбитальные транспортные операции по доставке полезной нагрузки на рабочую орбиту. Преимущества ЭРД при решении этой задачи заключается в возможности высокоточного формирования заданных рабочих орбит спутников Земли и в доставке на рабочую орбиту существенно большего полезного груза.

## 1.1 Конструктивно компоновочная схема американского проекта «SEPS»

Приведём примеры проектов электроракетных буксиров (ЭРБ) с СЭУ. В 70–е годы XX века в США был разработан проект универсальной космической ступени SEPS (рисунок 1) с солнечной электроракетной двигательной установкой, предназначавшейся для многочисленных межпланетных и межорбитальных полётов, таких как перелёты на геостационарную орбиту, встреча с кометой и астероидами, полёты к дальним планетам.

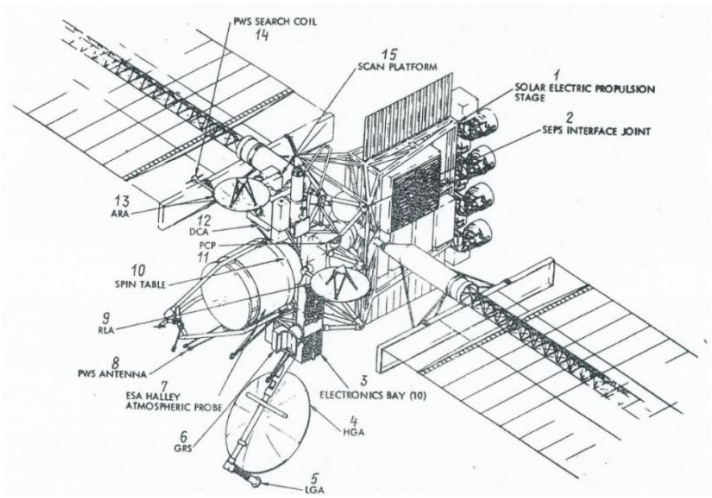



Рисунок 1 – Универсальная ступень SEPS

1– двигательная установка; 2 – узел стыковки; 3 – отсек электронной аппаратуры; 4 – остронаправленная антенна; 5 – всенаправленная антенна; 6 – блок гироприборов; 7 – исследовательский зонд; 8 – радиокоммуникационная антенна зонда; 9 – антенна радиолокатора; 10 – платформа для закрутки зонда; 11 – энергоблок; 12 – ОСА; 13 – радиолокационная антенна; 14 – радиопоисковое устройство системы связи с зондом; 15 – платформа с приборами обзора.

Исследований показывают, что при равных мощностях энергоустановок, электроракетные буксиры с солнечной энергоустановкой (СЭУ) лидируют по величине годового грузопотока при доставке грузов на высокие орбиты, например, такие, как геостационарная орбита (ГСО) и орбиты искусственного спутника Луны (ОИСЛ) [4].

## 1.2 Конструктивно компоновочная схема проекта МТА РКК «Энергия»

В настоящее время технический прогресс в создании солнечных батарей, использующих ФЭП на арсениде галлия или аморфном кремнии, и средств выведения (РН семейства «Ангара») позволяет создать и вывести на орбиту электроракетный буксир с СЭУ мощностью до 400 кВт.

Одним из последних проектов межорбитальных буксиров с солнечной ЭРДУ является разработанный в РКК «Энергия» электроракетный буксир с солнечной энергоустановкой мощностью 400 кВт (рисунок 2).

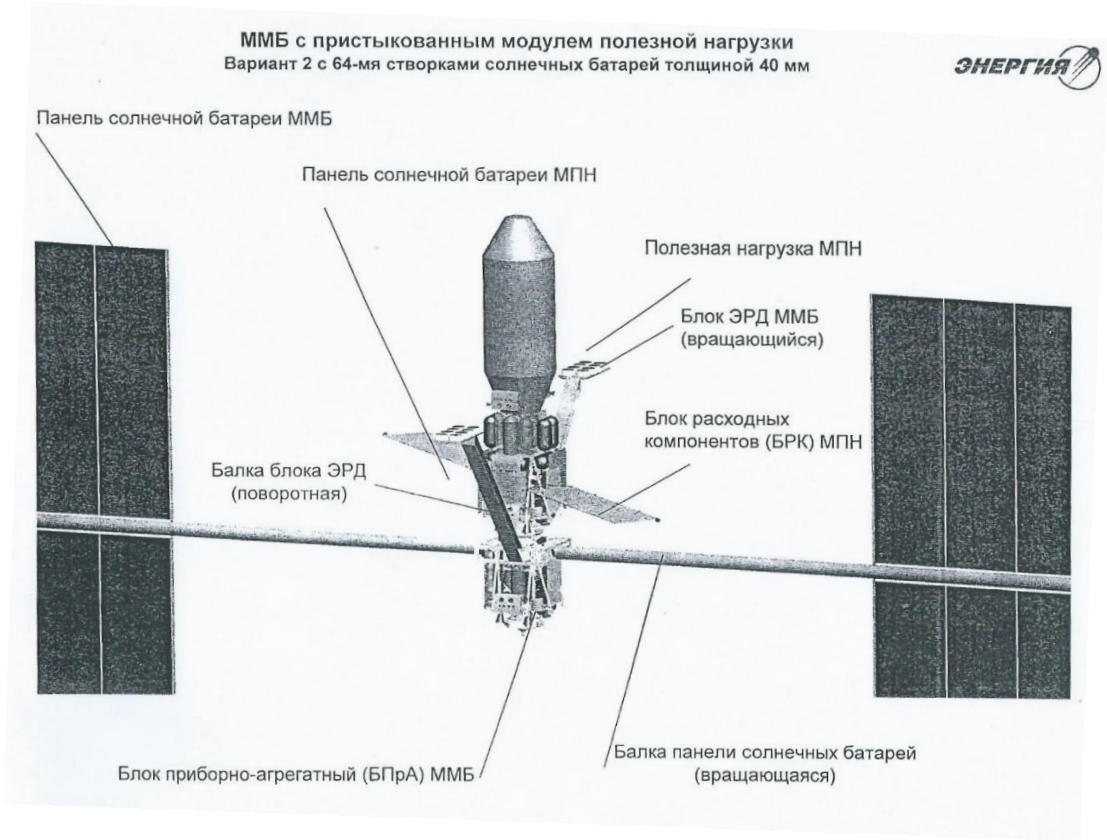



Рисунок 2 – Многоразовый межорбитальный буксир (ММБ) с электроракетной двигательной установкой и с солнечной энергоустановкой мощностью 400 кВт

### 1.3 Основные системы МТА

Важнейшими параметрами систем МТА являются масса, объем, энергопотребление, аппаратурное решение, надежность, потребное резервирование и стоимость. Основные критерии при выборе систем традиционные, включающие малую стоимость, гибкость в использовании, длительный срок эксплуатации, высокую надежность и возможно большую унификацию. При выборе систем следует помнить о некоторых общих закономерностях проектирования космических объектов.

На выбор компонентов систем часто накладываются ограничения внешнего характера, например, располагаемые обводы, наличие антенн, сопл реактивной системы ориентации и стабилизации, солнечных батарей, радиаторов и пр. Многие системы могут потребовать наличия устройств развертывания и приведения их в рабочее состояние.

Следует учитывать взаимовлияние отдельных подсистем и их характеристик, когда незначительные изменения параметров одной системы (массы, энергопотребления, объема, стоимости) приводит к изменению общей массы, энергопотребления и стоимости МТА в целом.

Определение состава некоторых систем, таких, как системы навигации, управления, связи, радиотелеметрическая, является критическим моментом для работоспособности всего МТА.

При включении в состав МТА перспективных систем необходимо учитывать их стоимость, а также время на разработку и поставку оборудования для их производства.

Для более точной оценки эффективности каждой системы необходимо рассматривать ее возможности при работе в составе пилотируемого и беспилотного МТА, в многоразовых и одноразовых конструкциях, в вариантах наземного и космического базирования МТА.

Предварительные оценки массы МТА показывают, что увеличение массы любой системы МТА на 1 кг эквивалентно увеличению стартовой массы МТА для полета на геосинхронную орбиту на 7..12 кг, а увеличение энергопотребления на 1 Вт в течение 7 сут. полета приводит к увеличению стартовой массы на 2,5..4,5 кг.

Перечислим основные системы МТА, которые являются общими как для МТА с химическими двигателями, так и для МТА с ЭРДУ:

1) Система наведения, навигации и управления;
2) Система связи и обработки информации;
3) Система энергопитания;
4) Система терморегуляции;
5) Система исполнительных органов системы ориентации и стабилизации.

## 2 Проектно–баллистический анализ МТА с ЭРДУ

Рассмотрим задачу проектно–баллистической оптимизации МТА с СЭРДУ. В общем случае, целью МТА является доставка полезной нагрузки (ПН) на заданную рабочую орбиту и возвращение МТА на исходную орбиту. Обычно требуется многократное повторение таких маневров, возможно с отличающимися параметрами ПН или рабочих орбит. Оптимизация траекторий управляемого движения и проектного облика МТА должна обеспечивать наибольшую массу полезной нагрузки при фиксированной стартовой массе МТА.

### 2.1 Проектно–баллистический расчёт межорбитального перехода между некомпланарными круговыми орбитами

Основной задачей расчета является расчет межорбитального перехода между некомпланарными круговыми орбитами.

Исходная орбита – низкая круговая орбита H = 300 км, i = 51,6°.

Конечная орбита: R = 42164 км, i = 0°.

Предполагается, что продолжительность полета является варьируемой величиной в диапазоне от 100 суток до 400 суток.

Предполагается, что дата старта выбирается оптимальной с учетом предварительно сделанных расчетов так, чтобы космический аппарат (КА) во время перелета не заходил в тень.

Проектно–баллистический расчет траектории сводится к решению дифференциальных уравнений в оскулирующих элементах.

Система дифференциальных уравнений в оскулирующих элементах:

$$
\begin{cases}
\dfrac{dA}{dt} = \dfrac{2p}{(1-e)^2}\sqrt{\dfrac{p}{\mu}} \cdot \left[e\sin\vartheta \cdot S + (1+e\cos\vartheta)\cdot T\right], \\
\dfrac{de}{dt} = \sqrt{\dfrac{p}{\mu}} \cdot \left[\sin\vartheta \cdot S + \dfrac{e\cos^2\vartheta + 2\cos\vartheta + e}{1+e\cos\vartheta}\cdot T\right], \\
\dfrac{di}{dt} = \sqrt{\dfrac{p}{\mu}} \cdot \dfrac{\cos u}{1+e\cos\vartheta}\cdot W, \\
\dfrac{d\omega}{dt} = \dfrac{1}{e}\sqrt{\dfrac{p}{\mu}} \cdot \left[-\cos\vartheta \cdot S + \dfrac{\sin\vartheta(2+e\cos\vartheta)}{1+e\cos\vartheta}\cdot T - \dfrac{e\sin u \cdot ctgi}{1+e\cos\vartheta}\cdot W\right], \\
\dfrac{d\Omega}{dt} = \sqrt{\dfrac{p}{\mu}} \cdot \dfrac{\sin u}{\sin i(1+e\cos\vartheta)}\cdot W, \\
\dfrac{du}{dt} = \dfrac{\sqrt{\mu p}}{p^2} \cdot \left[(1+e\cos\vartheta)^2 - \dfrac{p^2}{(1+e\cos\vartheta)\mu}\cdot ctgi \cdot \sin u \cdot W\right],
\end{cases}
$$

где $p = A(1-e^2)$ – фокальный параметр;

$\vartheta = u - \omega$ – истинная аномалия;

$e$ – эксцентриситет;

$\omega$ – угловое ускорение перицентра от узла;

$\Omega$ – долгота восходящего узла;

$i$ – наклонение орбиты;

$\tau$ – время прохождения через перицентр;

$t$ – время; $\vartheta$ – истинная аномалия;

$u$ – аргумент широты;

$S,T,W$ – проекции реактивного ускорения на направление радиус–вектора, на перпендикулярное к нему в плоскости орбиты и на перпендикулярное к плоскости орбиты;

$\mu = fM$ – произведение гравитационной константы на массу притягивающего центра.

На рисунках 3,4 показаны две правые системы координат: орбитальная (*Onrb)* и связанная с КА (*OXYZ*). Вектор тяги $\bar{P}$ направлен вдоль оси *OX*.

Выражения для компонент реактивного ускорения в орбитальной системе координат:

$$T = \delta a \cos\lambda \cos\psi,$$
$$S = \delta a \sin\lambda \cos\psi,$$
$$W = \delta a \sin\psi,$$

где $a$ – модуль полного реактивного ускорения $\left(a = \dfrac{a_0}{1 - a_0 t / c}\right)$;

$\delta$ – функция включения–выключения двигателей $\left(\delta = \{0,1\}\right)$;

$\lambda$ – угол ориентации вектора тяги в плоскости орбиты $\left(\lambda \in \left[0^o; 180^o\right]\right)$;

$\psi$ – угол ориентации вектора тяги в плоскости местного горизонта $(\psi \in \left[-90^o; 90^o\right])$.

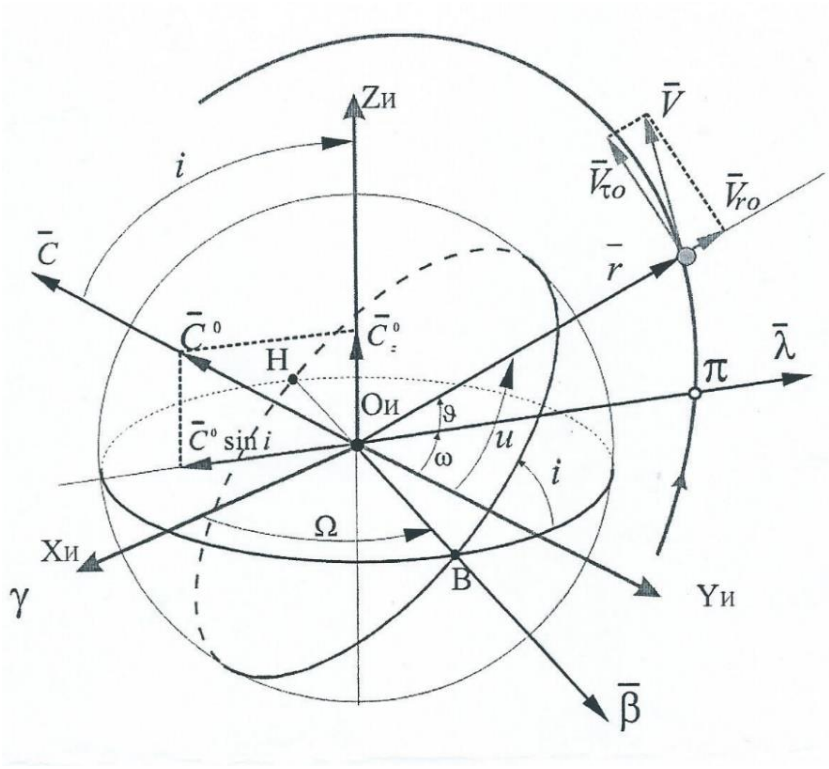



Рисунок 3 – К дифференциальным уравнения в оскулирующих элементах

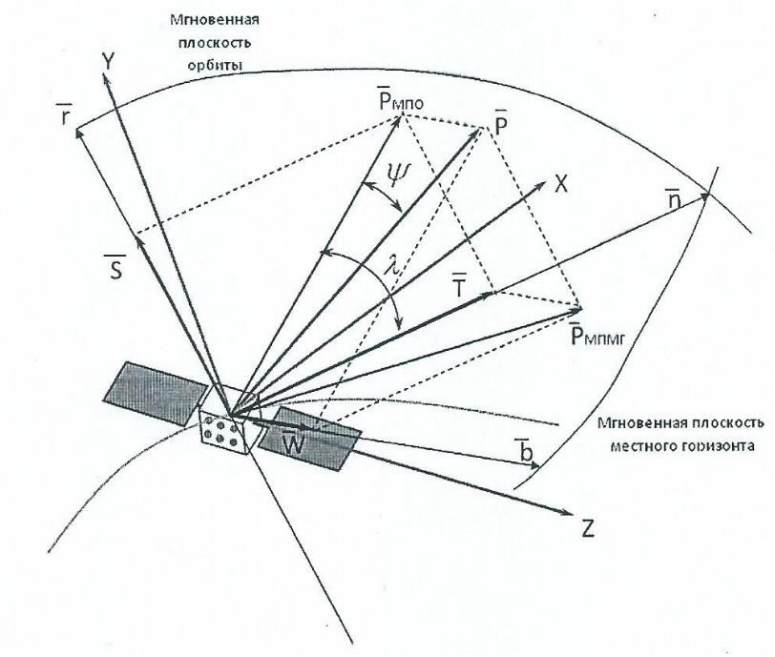



Рисунок 4 – Орбитальная и связанная СК

Дифференциальные уравнения в оскулирующих элементах могут быть решены аналитическим методом или с помощью численных методов (например, с помощью метода Рунге–Кутты 4–го порядка) [2].

2.2 Проектная модель МТА с ЭРДУ

Проектная модель описывает массу МТА с СЭРДУ как сумму масс следующих основных компонентов:

$$M_0 = M_{ПН} + M_Э + M_Д + M_{РТ} + M_{СПХ} + M_К, \quad (1)$$

где $M_{ПН}$ – масса полезного груза;

$M_Э$ – масса энергоустановки, состоящая из источника и преобразователя энергии;

$M_Д$ – масса двигательной установка, включающая маршевые и управляющие двигатели вместе с исполнительными органами;

$M_{РТ}$ – масса рабочего тела, необходимого для прямого и обратного перелета с учетом расхода на управление;

$M_{СПХ}$ – масса системы подачи и хранения рабочего тела (баки, трубопроводы и пр.);

$M_К$ – корпус и конструкции МТА.

Наиболее простыми и часто использующимися зависимостями массы отдельных компонентов КА от проектных параметров являются линейные зависимости масс от номинальной мощности энергоустановки $N$ и тяги двигателей на стартовой орбите $P$:

$$M_{\text{Э}} = \alpha_{\text{Э}} \cdot N,$$

$$M_{\text{Д}} = \gamma_{\text{Д}} \cdot P,$$

$$M_{\text{К}} = \alpha_{\text{К}} \cdot M_0,$$

$$M_{\text{СПХ}} = k_{\text{СПХ}} \cdot M_{\text{РТ}},$$

где $\alpha_{\text{Э}}$ – удельная массовая характеристика энергоустановки, кг/Вт;

$\alpha_{\text{К}}$ – удельная масса конструкции по начальной массе, кг/Вт;

$\gamma_{\text{Д}}$ – удельная масса двигательной установки, кг/Н;

$P$ – тяга ЭРДУ, Н;

$k_{\text{СПХ}}$ – отношение массы системы подачи и хранения рабочего тела к массе рабочего тела.

Количество двигателей определяем в следующем порядке:

1) Определяем затраты характеристической скорости для перелета

$$V_{\text{хк}} = \sqrt{\frac{\mu}{r_0}} \cdot \sqrt{1 - 2\sqrt{\frac{r_0}{r_{\text{к}}}} \cdot \cos\frac{\pi \cdot (i_{\text{к}} - i_0)}{2} + \frac{r_0}{r_{\text{к}}}}, \qquad (*)$$

где $\mu = 398600 \cdot 10^9 \ \frac{\text{м}^3}{\text{с}^2}$ – гравитационный параметр Земли,

$r_0$ – радиус начальной орбиты;

$r_{\text{к}}$ – радиус конечной орбиты;

$i_0$ – наклонение начальной орбиты;

$i_{\text{к}}$ – наклонение конечной орбиты.

2) Рассчитываем тягу ЭРДУ

$$P = M_0 \cdot \frac{c}{T} \cdot \left(1 - e^{-\frac{V_{\text{хк}}}{c}}\right),$$

где $c$ – удельный импульс двигателя;

$T$ – время перелёта.

3) Определяем потребное количество двигателей

$$n = Trunc\left(\frac{P}{P_0}\right) + 1,$$

где $Trunc(x) -$ функция возвращающая ближайшее меньшее целое к $x$;

$P_0 -$ тяга одного двигателя.

Масса израсходованного рабочего тела на перелет выражается через моторное время или характеристическую скорость перелета

$$M_{\text{РТ}} = \frac{P}{c}T = M_0\left(1 - e^{-\frac{V_{\text{хк}}}{c}}\right).$$

Подставляя эти выражения в формулу (1) с учетом зависимости мощности от проектных параметров с учетом КПД

$$N = \frac{Pc}{2\eta}.$$

Получим новый вид уравнения баланса масс на начальной орбите:

$$M_{\text{ПН}} = M_0 - \alpha_{\text{Э}} \cdot \frac{Pc}{2\eta} - \gamma_{\text{Д}} \cdot P - \frac{P}{c}T_{\text{м}}(1 + k_{\text{СПХ}}) - \alpha_{\text{К}} \cdot M_0.$$

Оптимальная программа разворота вектора тяги относительно оскулирующей плоскости орбиты:

$$\psi(V_x, u) = arctg(tg\,\psi_m cos\,u),$$

где $\psi$ – угол ориентации тяги относительно плоскости орбиты;

$\psi_m$ – амплитуда периодических колебаний этого угла, зависящая от текущей характеристической скорости;

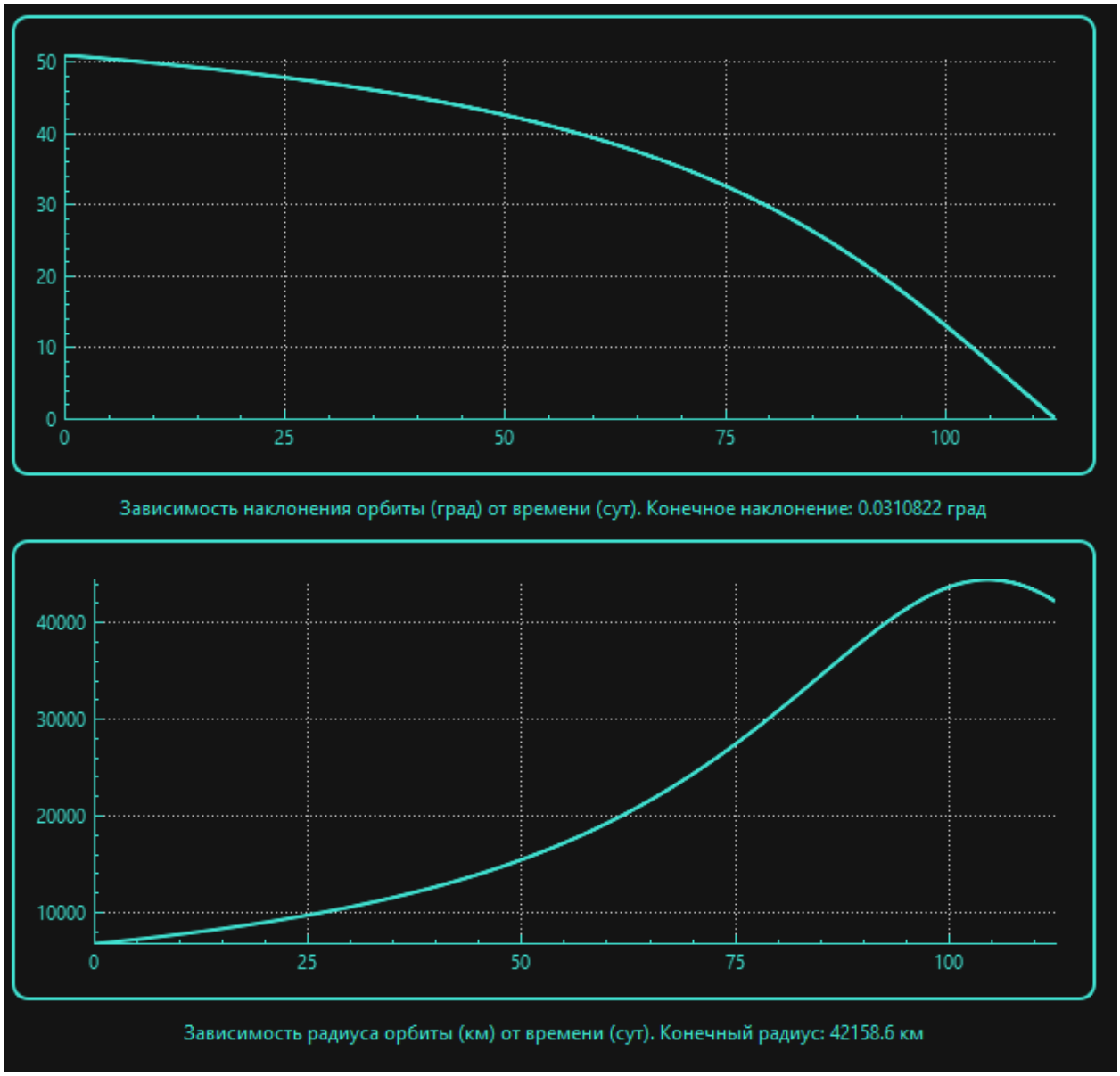
$u$ – аргумент широты.

$$\psi(V_x, u) = \psi_m(V_x) sign(cos\,u),$$

$$\psi_m = arctg\left\{\frac{\sin\pi\frac{(i_{\text{к}}-i_0)}{2}}{\sqrt{\frac{r_{\text{к}}}{r_0}}}\left[1 - \frac{\cos\pi\frac{(i_{\text{к}}-i_0)}{2}}{\sqrt{\frac{r_{\text{к}}}{r_0}}} - \frac{V_x}{V_0}\sqrt{1 - \frac{2\cos\pi\frac{(i_{\text{к}}-i_0)}{2}}{\sqrt{\frac{r_{\text{к}}}{r_0}}} + \frac{r_0}{r_{\text{к}}}}\right]^{-1}\right\}, (**)$$

$$i = i_0 - \frac{2}{\pi} arctg\{AV_x[C - (1 - B) \cdot V_x^2]^{-1}\},$$

$$A = \frac{\sin \pi \frac{(i_к - i_0)}{2}}{\sqrt{\frac{r_к}{r_0}}}, B = \frac{\cos \pi \frac{(i_к - i_0)}{2}}{\sqrt{\frac{r_к}{r_0}}}, C = \frac{\sqrt{1 - 2\sqrt{\frac{r_к}{r_0}} \cos \pi \frac{(i_к - i_0)}{2} + \frac{r_0}{r_к}}}{V_{xk}},$$




$$r = \left(\frac{\sin^2 \frac{\pi \cdot i_к}{2}}{r_к}\right)^{-1} \left(1 - \frac{2\cos \frac{\pi \cdot i_к}{2}}{\sqrt{r_к}} + \frac{1}{r_к}\right) (***).$$

Рисунок 5 – Зависимость радиуса орбиты и наклонения орбиты от характеристической скорости

Для сравнения результатов расчета при ручном режиме и при оптимизации расчета, путем выбора ЭРД из базы данных построим графики зависимостей относительной массы полезной нагрузки и мощности ЭУ от времени, используя следующие формулы:

1. Оптимальная скорость истечения для ручного режима:

$$c_{opt} = \sqrt{\frac{2 \cdot T_M \cdot \eta}{\alpha}},$$

где $T$ – время перелёта, с;

$\eta$ – КПД;

$\alpha_э$ – удельная массовая характеристика энергоустановки, кг/Вт.

2. Относительная масса полезной нагрузки:

$$\mu_{ПН} = \frac{M_{ПН}}{M0},$$

где $M_{ПН}$ – масса полезной нагрузки, кг;

$M0$ – начальная масса, кг.

Расчеты проводятся для различной продолжительности прямого перелета ($T$=150; 175; 200; 225; 250; 275; 300 суток).

3 Описание программы

Программа «Расчет основных проектных параметров» написана на языке программирования Lazarus, а построение 3D модели КА в SolidWorks2019 написана на языке Delphi 7, коды программ представлены в дополнении А.

3.1 Основная форма

Внешний вид основной формы программы «Расчет основных проектных параметров» представлен на рисунке 6.

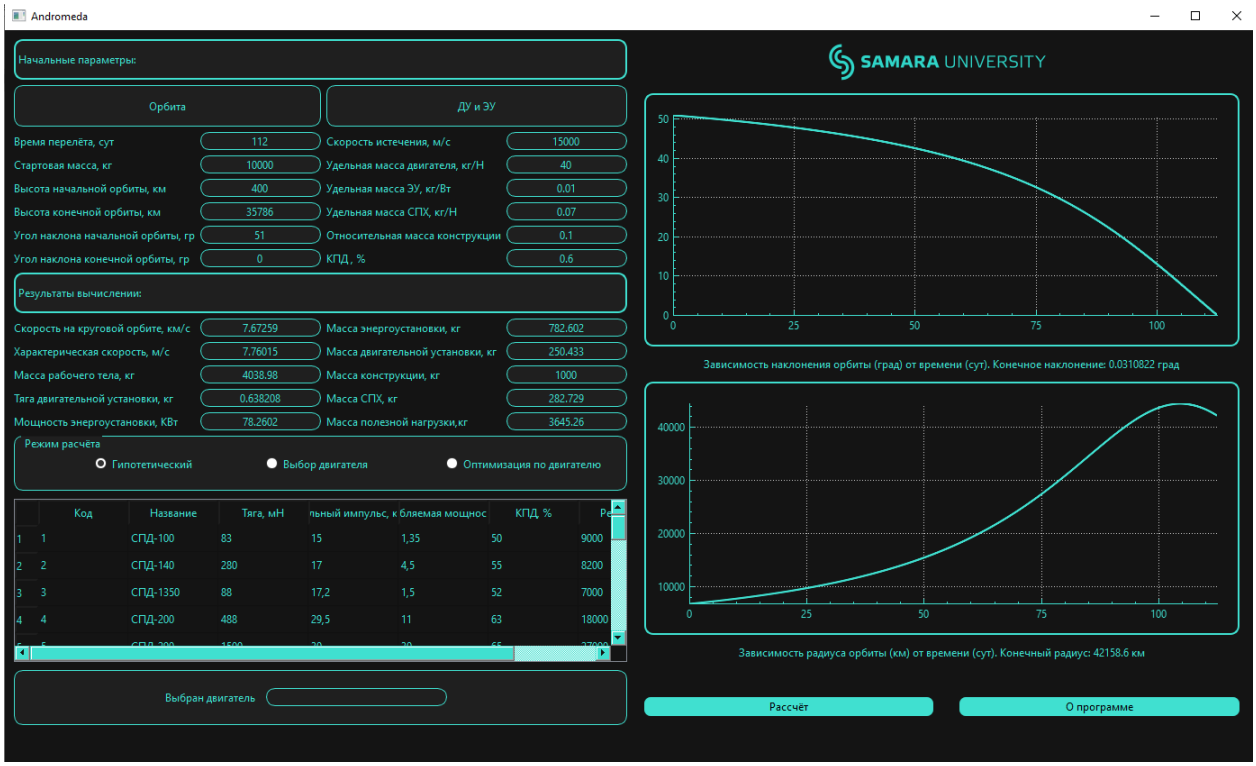



Рисунок 6 – Основная форма

Условно основную форму можно разделить на четыре блока.

Блок ввода основных исходных данных необходимых для решения задачи. На рисунке 7, на котором представлен данный блок, введены уже некие условные данные, что было необходимо для удобства отладки программы.

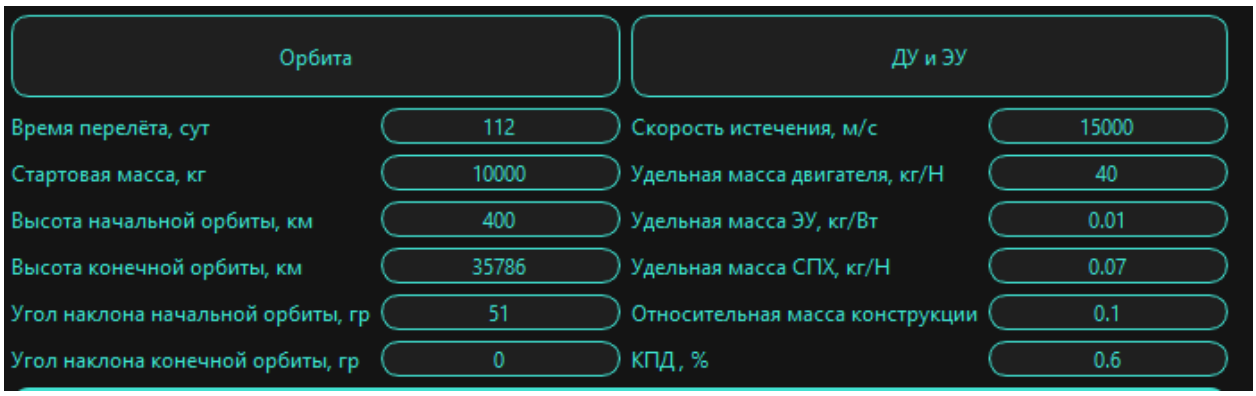



Рисунок 7 – Блок ввода исходных данных

Блок вывода результатов расчета (Рисунок 8). Здесь выводятся результаты после первого расчета, производимого нажатием кнопки «Рассчитать».

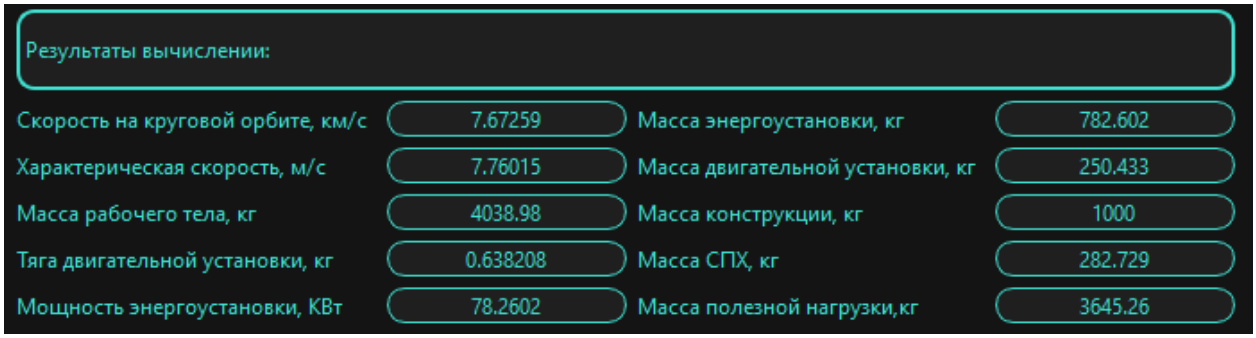



Рисунок 8 – Блок вывода численных результатов

Блок выведения графических результатов (Рисунок 9). Здесь выводятся графики зависимостей радиуса и наклонения орбиты в зависимости от времени. Оба графика строятся по аналитически полученным решениям системы дифференциальных уравнений движения методом Рунге-Кутты.

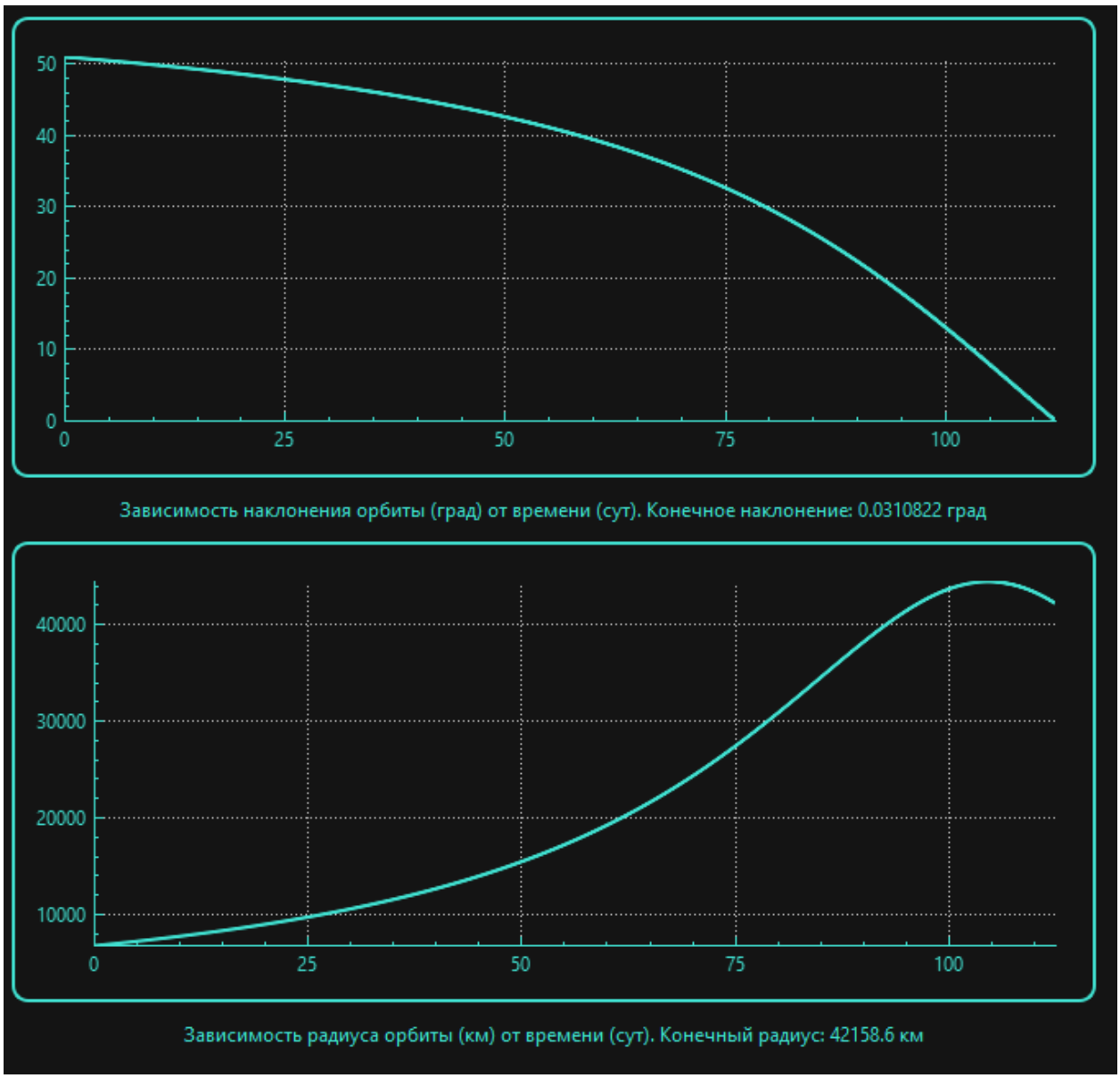



Рисунок 9 – Блок вывода графических результатов

3.2 Работа с базой данных

К форме подключена база данных ЭРДУ. С помощью окна «Выбор режима» можно выбрать режим расчета, а именно: «Ручной», «Выбор ЭРД из БД», «Оптимизация». Далее нажатием кнопки «Рассчитать» выводятся проектные параметры, рассчитанные для ЭРДУ. Внешний вид окна представлен на рисунке 10.


Рисунок 10 – Расчет массовых характеристик

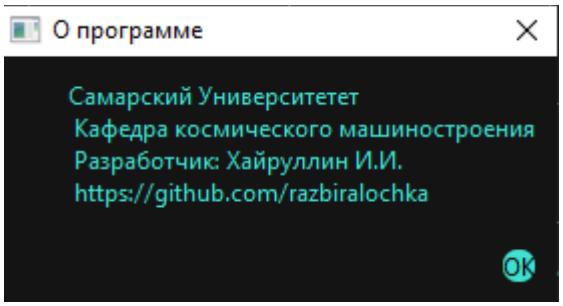



Рисунок 11 – Кнопка «О программе»

### 3.3 Работа с пакетом Solid Works

При нажатии кнопки «Построить 3D модель» открывается пакет Solid Works и в нем строится модель МТА (рисунок 14). При нажатии кнопки «Выйти из SolidWorks 2019» пакет Solid Works закрывается. Внешний вид окна представлен на рисунке 13.

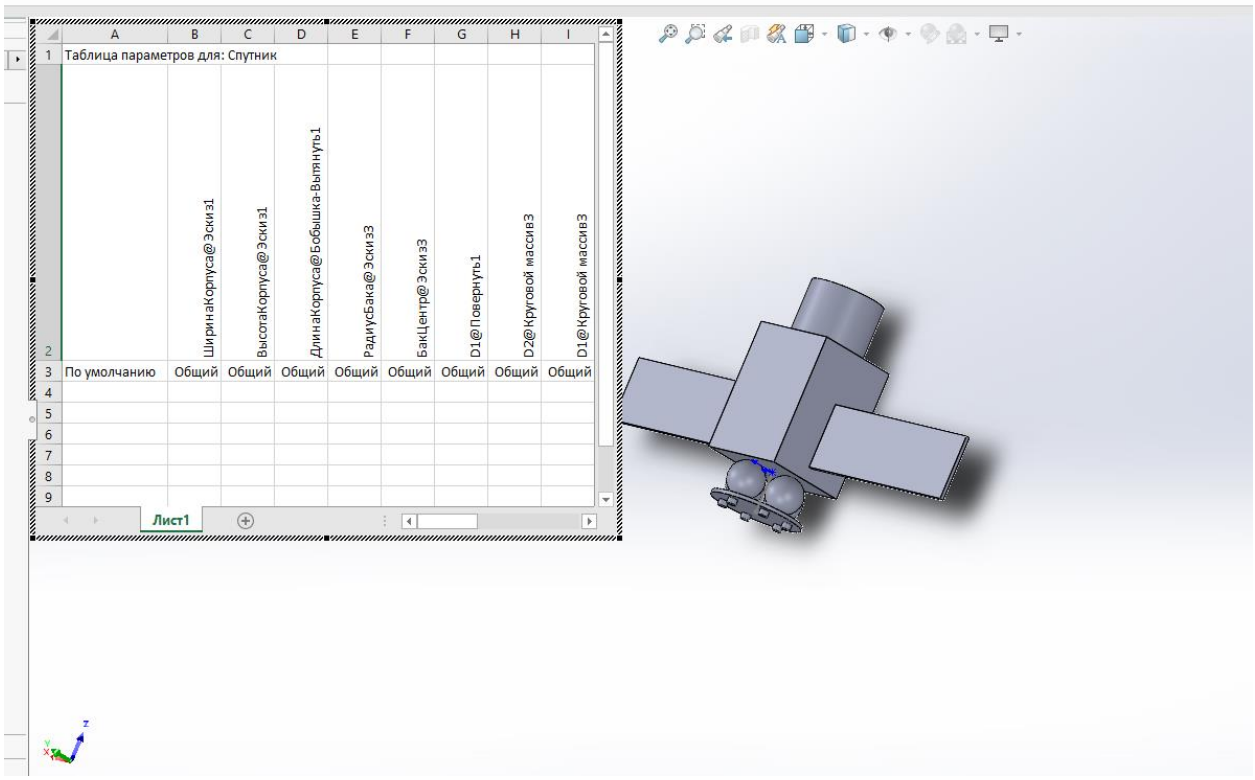



Рисунок 12 – Работа с пакетом Solid Works

## 4 Расчет проектно-баллистических параметров МТА

### 4.1 Расчет оптимальных параметров МТА. Расчет оптимальных проектно-баллистических параметров МТА с учетом характеристик, существующих ЭРД

Вычислим оптимальные проектно-баллистические параметры, задав время полета 150,175, 200, 225, 250, 275, 300 суток. Будем использовать следующие исходные данные при поиске оптимальных проектных параметров: $a_э$= 10 кг/кВт, $a_{пэ}$ = 5 кг/кВт, $y_{спх}$ = 0.07, $h_к$=0,1. Параметры двигателей будем выбирать из таблицы. Стартовая орбита — круговая с высотой М = 6771 км и наклонение 51. Стартовая масса аппарата равна Мо, = 7000 кг. В качестве критерия оптимальности выберем относительную массу полезной нагрузки. Решение задачи выбора оптимальных параметров будем производить с помощью программы, разработанной в интегрированной среде Lazarus. Результаты расчета оптимальных проектных параметров МТА для

различных времен перелета приведены в таблице 1 и на рисунке 15. Из полученных результатов видно, что относительная масса полезной нагрузки существенно возрастает с увеличением времени перелета.

Вычислим проектно-баллистические параметры МТА с учетом характеристик существующих ЭРД, задав время полета 150,175, 200, 225, 250, 275, 300 суток и объединим их на графиках.

Таблица 1 Результаты решения задачи оптимизации для различных значений времени полета

| $M_{пн}$,кг | Модель РД | n | P, H | N, kBt | T |
|---|---|---|---|---|---|
| 4688 | RIT-XT | 19 | 4,14 | 119,7 | 150 |
| 4903 | RIT-XT | 16 | 3,43 | 100,8 | 175 |
| 5047 | RIT-XT | 14 | 3,05 | 88,7 | 200 |
| 5119 | RIT-XT | 13 | 2,84 | 81,9 | 225 |
| 5262 | RIT-XT | 11 | 2,39 | 69,3 | 250 |
| 5334 | RIT-XT | 10 | 2,18 | 63 | 275 |
| 5334 | RIT-XT | 10 | 2,18 | 63 | 300 |

На рисунке 15 представлена зависимость массы полезной нагрузки (ось y) от времени (ось x). Красный график-гипотетический расчет, в котором

использовался 1 гипотетический двигатель. Синий график – режим оптимизации, в котором из базы данных двигателей выбирался оптимальный. Критерий оптимальности- масса полезной нагрузки на борту КА.

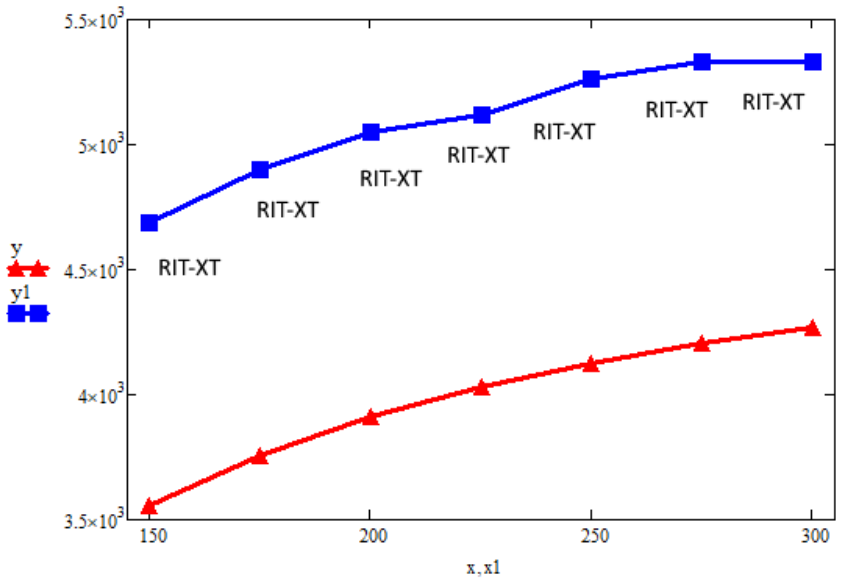



Рисунок 13 – График зависимости массы полезной нагрузки от времени

На рисунке 16 представлена зависимость мощности энергетической установки (ЭУ) (ось y) от времени (ось x). Красный график-гипотетический расчет, в котором использовался 1 гипотетический двигатель. Синий график – режим оптимизации, в котором из базы данных двигателей выбирался оптимальный. Критерий оптимальности- масса полезной нагрузки на борту КА.

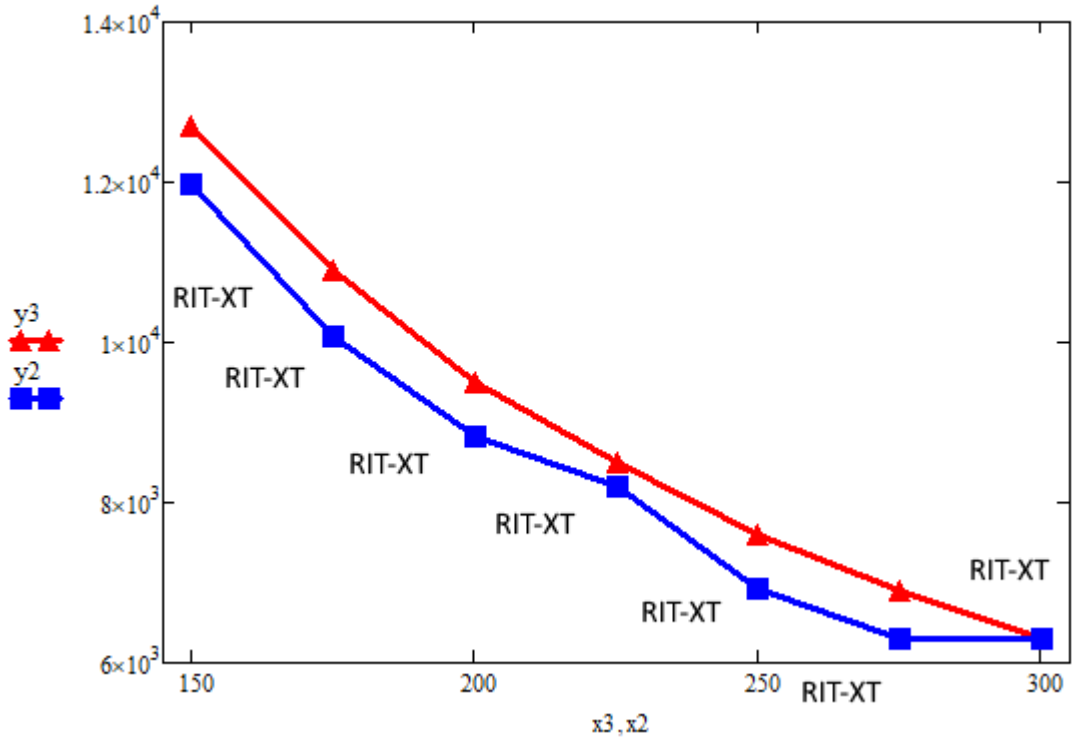



Рисунок 14 – График зависимости мощности ЭУ от времени

4.2 Модель, построенная с использованием пакета Solid Works

Вычислим проектно-баллистические параметры МТА с учетом характеристик, существующих ЭРД, задав время полета 150,175, 200, 225, 250, 275, 300 суток.

Модель, построенная с использованием пакета Solid Works для времени перелета равным 150 суток, показана на рисунке 17.

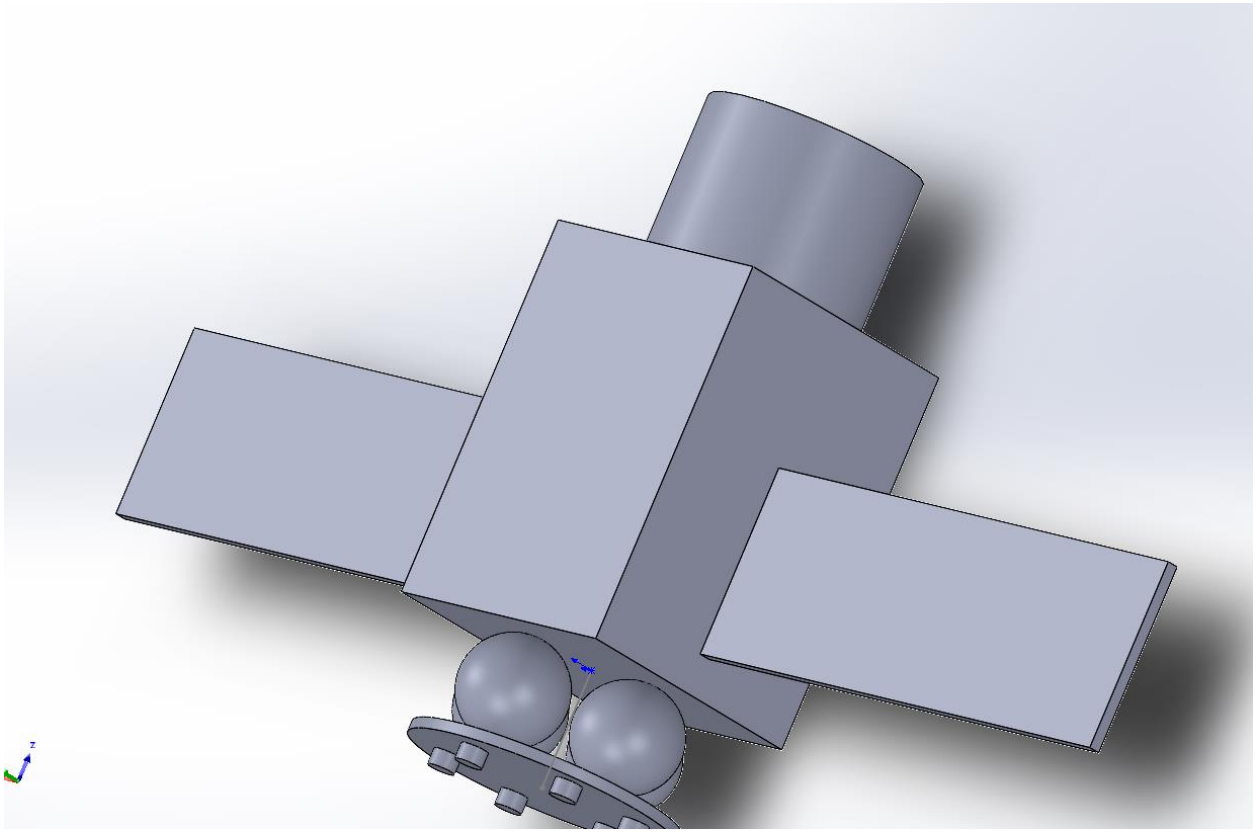


Рисунок 15 – Проектный облик МТА в пакете Solid Works (T=150 суток)

Выберем для межорбитального полёта третий вариант многоразового МТА с продолжительность перелёта 150 суток.

Основные проектные параметры выбранного варианта многоразового МТА представлены в таблице 2.

Таблица 2 Основные проектные параметры многоразового МТА (Т=150 сут)

| Параметр | Значение |
|---|---|
| Масса полезной нагрузки, кг | 4688 |
| Стартовая масса, кг | 7000 |
| Модель двигателя | RIT-XT |
| Тяга одного двигателя, Н | 0.218 |
| Количество двигателей | 19 |
| Тяга ЭРДУ, Н | 4,142 |
| Мощность энергоустановки, кВт | 119700 |
| Начальное ускорение, м/с^2 | 0,00059 |
| Длительность перелёта, сутки | 150 |
| Масса энергоустановки, кг | 1197 |
| Масса СПХ, кг | 80 |
| Масса двигательной установки, кг | 166 |
| Масса конструкции, кг | 70 |
| Площадь солнечных батарей, м” | 318 |
| Масса рабочего тела, кг | 799 |

# ЗАКЛЮЧЕНИЕ

Рассмотрены теоритические вопросы оптимизации проектно-баллистических параметров МТА, предназначенного для перелетов между некомпланарными околоземными орбитами. Задача оптимизации разбивается на две задачи, а именно динамическую и параметрическую. В качестве решения динамической задачи используются решения, полученные в [1] и отраженные в данной пояснительной записки формулами (*), (**), (***).

Разработано программное обеспечение, позволяющее выполнить расчет и моделирование МТА с ЭРД с заданными начальными условиями. Данное программное обеспечение помогает автоматизировать процесс выбора оптимальных проектных параметров и сформировать проектный облик аппарата и отдельных его систем. Кроме того, к программе подключена база данных, содержащая информацию о различных ЭРД. Для этих двигателей программа выполняет расчет их количества в двигательной установке, определяет массы систем космического аппарата. Так же в программе реализован алгоритм поиска двигателя, обеспечивающего вывод максимальной полезной нагрузки на заданную высоту. Использование системы твердотельного моделирования «Solid Works» позволяет осуществить детализацию проектного облика аппарата и отдельных его систем. Использование этой методики позволяет осуществить расчет большого количества проектных вариантов.

Проведен расчет проектно-баллистических параметров МТА с ЭРД с использованием программного обеспечения для осуществления продолжительности перелетов 150, 175, 200, 225, 250, 275, 300 суток. При заданных условиях оптимальным является двигатель RIT-XT.

В результате серии расчетов было установлено, что с увеличением времени полета МТА уменьшается потребная тяга и как следствие необходимая мощность энергоустановки и количество двигателей, увеличивается масса полезной нагрузки

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1 Лебедев, В.Н. Расчет движения космического аппарата с малой тягой [Текст] / Лебедев В.Н. – Москва: ВЦ АН СССР, 1968. – 108 с.

2 Салмин, В. В. Методы решения вариационных задач механики космического полёта с малой тягой [Текст] / Салмин В. В., Ишков С. А., Старинова О. Л. – Самара: Издательство Самарского научного центра РАН, 2006. – 164 с.

3 Салмин, В. В. Выбор основных проектных характеристик и конструктивного облика межорбитальных транспортных аппаратов с электрореактивными двигательными установками с использованием системы Solid Works [Текст]: учеб. пособие / В. В. Салмин, С. А. Ишков, О.Л.Старинова. – Самара: Изд-во Самар.гос.аэрокосм. ун-та, 2006. – 82 с.

4 Научно-технический отчет по ТЗ РКК «Энергия». СГАУ, НИИ космического машиностроения, 2013.

ПРИЛОЖЕНИЕ А

Файл "main.cpp"

```
#include "mainwindow.h"
#include <QApplication>
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

Файл "mainwindow.cpp"

```
#include "Initialize.h"
#include "RungeKutta.h"
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "Modeling.h"
#include <QMessageBox>
#include <math.h>

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    db = QSqlDatabase::addDatabase("QODBC");
    db.setDatabaseName("DRIVER={Microsoft Access Driver (*.mdb,
*.accdb)};FIL={MS Access};DSN='';DBQ=.//database/engine.mdb");


    qDebug() << db.drivers();
    qDebug() << db.open();
    if(!db.open())
    {
```

```
        QMessageBox::critical(this, "Ошибка открытия Базы Данных",
db.lastError().text());

    }

    model = new QSqlTableModel(this,db);
    model->setTable("engine");
    model->select();
    ui->tableView->setModel(model);



    ui->widget->setBackground(QColor(20, 20, 20));
    ui->widget->xAxis->setBasePen(QPen(QColor(64,224,208), 1));
    ui->widget->yAxis->setBasePen(QPen(QColor(64,224,208), 1));
    ui->widget->xAxis->setTickLabelColor(QColor(64,224,208));
    ui->widget->yAxis->setTickLabelColor(QColor(64,224,208));
    ui->widget->xAxis->setBasePen(QPen(QColor(64,224,208), 1));
    ui->widget->yAxis->setBasePen(QPen(QColor(64,224,208), 1));
    ui->widget->xAxis->setTickPen(QPen(QColor(64,224,208), 1));
    ui->widget->yAxis->setTickPen(QPen(QColor(64,224,208), 1));
    ui->widget->xAxis->setSubTickPen(QPen(QColor(64,224,208), 1));
    ui->widget->yAxis->setSubTickPen(QPen(QColor(64,224,208), 1));
```

```
    ui->widget_4->setBackground(QColor(20, 20, 20));
    ui->widget_4->xAxis->setBasePen(QPen(QColor(64,224,208), 1));
    ui->widget_4->yAxis->setBasePen(QPen(QColor(64,224,208), 1));
    ui->widget_4->xAxis->setTickLabelColor(QColor(64,224,208));
    ui->widget_4->yAxis->setTickLabelColor(QColor(64,224,208));
    ui->widget_4->xAxis->setBasePen(QPen(QColor(64,224,208), 1));
    ui->widget_4->yAxis->setBasePen(QPen(QColor(64,224,208), 1));
    ui->widget_4->xAxis->setTickPen(QPen(QColor(64,224,208), 1));
    ui->widget_4->yAxis->setTickPen(QPen(QColor(64,224,208), 1));
    ui->widget_4->xAxis->setSubTickPen(QPen(QColor(64,224,208), 1));
    ui->widget_4->yAxis->setSubTickPen(QPen(QColor(64,224,208), 1));









}


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

```
void MainWindow::on_pushButton_2_clicked()
{
   QMessageBox::about(this, "О программе", "Самарский Университетет\n Кафедра космического машиностроения\n Разработчик: Хайруллин И.И.\n https://github.com/razbiralochka");
}


void MainWindow::on_pushButton_clicked()
{

      Fly_Time=ui->lineEdit->text().toDouble()*86400;
      Start_Orbit_Height=ui->lineEdit_4->text().toDouble();
      Start_Orbit_Inclination=ui->lineEdit_6->text().toDouble()*M_PI/180;
      Finally_Orbit_Height=ui->lineEdit_5->text().toDouble();
      Finally_Orbit_Inclination=ui->lineEdit_7->text().toDouble()*M_PI/180;
      Start_SC_Mass=ui->lineEdit_2->text().toDouble();
      Realitive_Construct_Mass=ui->lineEdit_12->text().toDouble();
      SSS_Realitive_Mass=ui->lineEdit_11->text().toDouble();
      Gas_Flow_Speed=ui->lineEdit_8->text().toDouble();
      Engine_Specific_Mass=ui->lineEdit_9->text().toDouble();
      Electro_Specific_Mass=ui->lineEdit_10->text().toDouble();
      EFFICIENCY=ui->lineEdit_3->text().toDouble();
```

```
    if (ui->radioButton->isChecked())
{

    SdelatRaz();
}



QSqlQuery query;

    if (ui->radioButton_2->isChecked())
{

        query.exec("SELECT * FROM engine WHERE Код = " +
SelectedEngine);
        while (query.next())
        {
          name = query.value(1).toString();
          Engines_Thrust = query.value(2).toDouble() * 0.001;
          Gas_Flow_Speed = query.value(3).toDouble() * 1000.0;
```

```
            Engines_Power =  query.value(4).toDouble() * 1000.0;
            EFFICIENCY = query.value(5).toDouble() * 0.01;
          }

          SdelatDva();
          ui->lineEdit_23->setText(name);



    }


    double MAXIMUM_PAYLOAD_MASS=0;

    if (ui->radioButton_3->isChecked())
{
        for (int i = 1; i<=18; i++)
        {
        query.exec("SELECT * FROM engine WHERE Код = " +
QString::number(i));
        while (query.next())
        {
          name = query.value(1).toString();
          Engines_Thrust = query.value(2).toDouble() * 0.001;
```

```
  Gas_Flow_Speed = query.value(3).toDouble() * 1000.0;
  Engines_Power =  query.value(4).toDouble() * 1000.0;
  EFFICIENCY = query.value(5).toDouble() * 0.01;
 }

 SdelatDva();
 if (Payload_Mass > MAXIMUM_PAYLOAD_MASS)
 {
    MAXIMUM_PAYLOAD_MASS = Payload_Mass;
    EngineNumber=i;
    ui->lineEdit_23->setText(name);
 }
 }


 query.exec("SELECT * FROM engine WHERE Код = " +
QString::number(EngineNumber));
 while (query.next())
 {
  name = query.value(1).toString();
  Engines_Thrust = query.value(2).toDouble() * 0.001;
  Gas_Flow_Speed = query.value(3).toDouble() * 1000.0;
  Engines_Power =  query.value(4).toDouble() * 1000.0;
  EFFICIENCY = query.value(5).toDouble() * 0.01;
```

```
    }
    SdelatDva();
    ui->lineEdit_23->setText(name);

}

    foo = (Gas_Mass)/(3*M_PI);
    Tank_Radius=100*round(pow(foo,1/3));
    Tank_CTR = round(Tank_Radius*1.5);
    Body_lenght=round(500*pow(Construct_Mass,1/3));
    SP_Square =0.5* Engines_Power/(1.3*0.29*0.866);
    Payload_R=round(sqrt(Payload_Mass/(0.02*1.5*M_PI)));
    EnginesCount=round(Engines_Thrust);
    ui->lineEdit_3->setText(QString::number(EFFICIENCY));
    ui->lineEdit_8->setText(QString::number(Gas_Flow_Speed));
    ui->lineEdit_18->setText(QString::number(Electro_Mass));
    ui->lineEdit_22->setText(QString::number(Payload_Mass));
    ui->lineEdit_21->setText(QString::number(SSS_Mass));
    ui->lineEdit_19->setText(QString::number(Engines_Mass));
    ui->lineEdit_20->setText(QString::number(Construct_Mass));
    ui->lineEdit_17->setText(QString::number(Engines_Power/1000));
    ui->lineEdit_16->setText(QString::number(Engines_Thrust/9.81));
    ui->lineEdit_15->setText(QString::number(Gas_Mass));
    ui->lineEdit_14->setText(QString::number(Delta_velocity/1000));
```

```
ui->lineEdit_13->setText(QString::number(Initial_Speed/1000));

RK_STEPS =Fly_Time/100;
INITIAL_ACCELERATION = Engines_Thrust/Start_SC_Mass;
ANGLE_U = 0;
RADIUS = Start_Orbit_Radius;
VELOCITY=0;
INCLINATION=Start_Orbit_Inclination;
TIME = 0;

QVector <double> RADIUS_ARRAY(0);
QVector <double> INCLINATION_ARRAY(0);
QVector <double> TIME_ARRAY(0);

Runge_Kutta(&TIME_ARRAY,&RADIUS_ARRAY,&INCLINATION_ARRAY
);
```

```
ui->widget->xAxis->setRange(0,Fly_Time/86000);
ui->widget->yAxis->setRange(180*Start_Orbit_Inclination/M_PI,180*Finally_Orbit_Inclination/M_PI);
ui->widget->addGraph();
ui->widget->graph(0)->setPen(QPen(QColor(64,224,208), 2));
ui->widget->graph(0)->addData(TIME_ARRAY,INCLINATION_ARRAY);
ui->widget->replot();
ui->widget_4->xAxis->setRange(0,Fly_Time/86000);
ui->widget_4->yAxis->setRange(Start_Orbit_Radius/1000.0,MAX_R/1000.0);
ui->widget_4->addGraph();
ui->widget_4->graph(0)->setPen(QPen(QColor(64,224,208), 2));
ui->widget_4->graph(0)->addData(TIME_ARRAY,RADIUS_ARRAY);
ui->widget_4->replot();

RADIUS_ARRAY.clear();
INCLINATION_ARRAY.clear();
TIME_ARRAY.clear();

ui->label_31->setText("Зависимость наклонения орбиты (град) от времени (сут). Конечное наклонение: " + QString::number(180*INCLINATION/M_PI) + " град");
ui->label_32->setText("Зависимость радиуса орбиты (км) от времени (сут). Конечный радиус: " + QString::number(RADIUS/1000)+ " км");
```

```
    ExcelExport();
    QMessageBox::about(this, "Excel Export", "Данные занесены в таблицу data.csv");
}


void MainWindow::on_tableView_activated(const QModelIndex &index)
{
   SelectedEngine = ui->tableView->model()->data(index).toString();
   ui->lineEdit_23->setText(name);
}
```

Файл "Initialize.h"

```
#ifndef INITIALIZE_H
#define INITIALIZE_H
#include "mainwindow.h"
#include <string>

#define Gravitation_Param 398600000000000

double Start_SC_Mass, Gas_Flow_Speed, Fly_Time, Start_Orbit_Height,
   Start_Orbit_Inclination,Finally_Orbit_Height,Finally_Orbit_Inclination,
Engine_Specific_Mass,
   Electro_Specific_Mass, SSS_Realitive_Mass, Realitive_Construct_Mass,
EFFICIENCY, Engines_Thrust,
   Start_Orbit_Radius, Finnaly_Orbit_Radius, Initial_Speed, Delta_velocity,
Engines_Power,Gas_Mass,
   Construct_Mass, Engines_Mass, Electro_Mass, SSS_Mass, Payload_Mass;

double TIME = 0;



__int8 EngineNumber = 1;
```

```
QString name;
QString SelectedEngine = "1";




void SdelatRaz()
{
   Start_Orbit_Radius = (6371 + Start_Orbit_Height)*1000;
   Finnaly_Orbit_Radius = (6371 + Finally_Orbit_Height)*1000;
   Initial_Speed = sqrt(Gravitation_Param/Start_Orbit_Radius);
   Delta_velocity = Initial_Speed*sqrt(1-
2*sqrt(Start_Orbit_Radius/Finnaly_Orbit_Radius)*cos((M_PI/2)*(Finally_Orbit_I
nclination-Start_Orbit_Inclination))+(Start_Orbit_Radius/Finnaly_Orbit_Radius));
   Gas_Mass = Start_SC_Mass*(1-exp((-Delta_velocity)/Gas_Flow_Speed));
   Engines_Thrust = (Gas_Flow_Speed*Gas_Mass)/(Fly_Time);
   Engines_Power = (Engines_Thrust*Gas_Flow_Speed)/(2*EFFICIENCY);
   Construct_Mass = Realitive_Construct_Mass * Start_SC_Mass;
   Engines_Mass = Engine_Specific_Mass * Engines_Thrust;
   Electro_Mass = Electro_Specific_Mass * Engines_Power;
   SSS_Mass = SSS_Realitive_Mass * Gas_Mass;
   Payload_Mass = Start_SC_Mass - Gas_Mass - Construct_Mass - SSS_Mass -
Engines_Mass - Electro_Mass;
}
```

```
void SdelatDva()
{
   Start_Orbit_Radius = (6371 + Start_Orbit_Height)*1000;
   Finnaly_Orbit_Radius = (6371 + Finally_Orbit_Height)*1000;
   Initial_Speed = sqrt(Gravitation_Param/Start_Orbit_Radius);
   Delta_velocity = Initial_Speed*sqrt(1-
2*sqrt(Start_Orbit_Radius/Finnaly_Orbit_Radius)*cos((M_PI/2)*(Finally_Orbit_I
nclination-Start_Orbit_Inclination))+(Start_Orbit_Radius/Finnaly_Orbit_Radius));
   Gas_Mass = Start_SC_Mass*(1-exp((-Delta_velocity)/Gas_Flow_Speed));


   Construct_Mass = Realitive_Construct_Mass * Start_SC_Mass;
   Engines_Mass = Engine_Specific_Mass * Engines_Thrust;
   Electro_Mass = Electro_Specific_Mass * Engines_Power;
   SSS_Mass = SSS_Realitive_Mass * Gas_Mass;
   Payload_Mass = Start_SC_Mass - Gas_Mass - Construct_Mass - SSS_Mass -
Engines_Mass - Electro_Mass;

}



int RK_STEPS =Fly_Time/100;
```

```
double INITIAL_ACCELERATION = Engines_Thrust/Start_SC_Mass;



double ANGLE_U = 0;

double RADIUS = Start_Orbit_Radius;

double VELOCITY=0;

double INCLINATION=Start_Orbit_Inclination;









#endif // INITIALIZE_H
```

Файл "RungeKutta.h"

```
#include "Initialize.h"
#include <math.h>
#include <QVector>
#ifndef RUNGEKUTTA_H
#define RUNGEKUTTA_H
#define Gravitation_Param 398600000000000



double sign(double ARG)
{
   if(ARG > 0) return 1;
   if(ARG == 0) return 0;
   if(ARG < 0) return -1;
}

double F_PSI()
{
   double U_ANG = ANGLE_U;
   double R_0 = Start_Orbit_Radius;
   double R_F = Finnaly_Orbit_Radius;
   double INC_0 = Start_Orbit_Inclination;
   double INC_F = Finally_Orbit_Inclination;
   double VEL_0 = Initial_Speed;
```

```
   double V_X = VELOCITY;
   double PSI_MAX = atan(sin(M_PI*(INC_F-INC_0)/2)/sqrt(R_F/R_0)*1/(1-
cos(M_PI*(INC_F-INC_0)/2)/sqrt(R_F/R_0)-V_X/VEL_0*sqrt(1-
2*cos(M_PI*(INC_F-INC_0)/2)/sqrt(R_F/R_0)+R_0/R_F)));

  if (INC_F < INC_0)
   {
      if (PSI_MAX < 0) return PSI_MAX*sign(cos(U_ANG));
      if (PSI_MAX > 0) return (PSI_MAX-M_PI)*sign(cos(U_ANG));
   }
   if (INC_F > INC_0)
   {
      if (PSI_MAX > 0) return PSI_MAX*sign(cos(U_ANG));
      if (PSI_MAX < 0) return (PSI_MAX+M_PI)*sign(cos(U_ANG));
   }



}

double X_ACC()
{

   return
INITIAL_ACCELERATION*cos(F_PSI())*exp(VELOCITY/Gas_Flow_Speed);
}
```

```
double Z_ACC()
{
   return -
INITIAL_ACCELERATION*sin(abs(F_PSI()))*exp(VELOCITY/Gas_Flow_Spe
ed)*sign(cos(ANGLE_U));//*cos(F_PSI())
}

double dr(double t,  double r, double inc, double u_ang, double vel)
{
   return 2*X_ACC()*sqrt(pow(r,3)/Gravitation_Param);
}

double di(double t,  double r, double inc, double u_ang, double vel)
{
   return Z_ACC()*sqrt(r/Gravitation_Param)*cos(u_ang);
}

double du(double t,  double r, double inc, double u_ang, double vel)
{
   return sqrt(Gravitation_Param/pow(r,3));
}

double dV(double t,  double r, double inc, double u_ang, double vel)
{
```

```
    return sqrt(pow(X_ACC(),2)+pow(Z_ACC(),2));
}

double k[4][4];
__int8 h = 100;
double MAX_R =0;

void Runge_Kutta(QVector<double> * LOCAL_TIME, QVector<double> *
LOCAL_RADIUS, QVector<double> * LOCAL_INCLINATION)
{

    for(int i = 0; i < RK_STEPS; i++)
    {
        k[0][0]= h * dr(TIME,RADIUS,INCLINATION, ANGLE_U, VELOCITY);
        k[1][0]= h * di(TIME,RADIUS,INCLINATION, ANGLE_U, VELOCITY);
        k[2][0]= h * du(TIME,RADIUS,INCLINATION, ANGLE_U, VELOCITY);
        k[3][0]= h * dV(TIME,RADIUS,INCLINATION, ANGLE_U, VELOCITY);

        k[0][1]= h *
dr(TIME+h/2.0,RADIUS+k[0][0]/2.0,INCLINATION+k[1][0]/2.0,
ANGLE_U+k[2][0]/2.0, VELOCITY+k[3][0]/2.0);
        k[1][1]= h *
di(TIME+h/2.0,RADIUS+k[0][0]/2.0,INCLINATION+k[1][0]/2.0,
ANGLE_U+k[2][0]/2.0, VELOCITY+k[3][0]/2.0);
```

```
k[2][1]= h *
du(TIME+h/2.0,RADIUS+k[0][0]/2.0,INCLINATION+k[1][0]/2.0,
ANGLE_U+k[2][0]/2.0, VELOCITY+k[3][0]/2.0);

k[3][1]= h *
dV(TIME+h/2.0,RADIUS+k[0][0]/2.0,INCLINATION+k[1][0]/2.0,
ANGLE_U+k[2][0]/2.0, VELOCITY+k[3][0]/2.0);


k[0][2]= h *
dr(TIME+h/2.0,RADIUS+k[0][1]/2.0,INCLINATION+k[1][1]/2.0,
ANGLE_U+k[2][1]/2.0, VELOCITY+k[3][1]/2.0);

k[1][2]= h *
di(TIME+h/2.0,RADIUS+k[0][1]/2.0,INCLINATION+k[1][1]/2.0,
ANGLE_U+k[2][1]/2.0, VELOCITY+k[3][1]/2.0);

k[2][2]= h *
du(TIME+h/2.0,RADIUS+k[0][1]/2.0,INCLINATION+k[1][1]/2.0,
ANGLE_U+k[2][1]/2.0, VELOCITY+k[3][1]/2.0);

k[3][2]= h *
dV(TIME+h/2.0,RADIUS+k[0][1]/2.0,INCLINATION+k[1][1]/2.0,
ANGLE_U+k[2][1]/2.0, VELOCITY+k[3][1]/2.0);


k[0][3]= h * dr(TIME+h, RADIUS+k[0][2],INCLINATION+k[1][2],
ANGLE_U+k[2][2], VELOCITY+k[3][2]);

k[1][3]= h * di(TIME+h, RADIUS+k[0][2],INCLINATION+k[1][2],
ANGLE_U+k[2][2], VELOCITY+k[3][2]);

k[2][3]= h * du(TIME+h, RADIUS+k[0][2],INCLINATION+k[1][2],
ANGLE_U+k[2][2], VELOCITY+k[3][2]);
```

```
        k[3][3]= h * dV(TIME+h, RADIUS+k[0][2],INCLINATION+k[1][2],
ANGLE_U+k[2][2], VELOCITY+k[3][2]);



        LOCAL_INCLINATION->push_back(180*INCLINATION/M_PI);
        LOCAL_RADIUS->push_back(RADIUS/1000.0);
        LOCAL_TIME->push_back(TIME/86000);

        TIME      += h;
        RADIUS    +=   (k[0][0]+2.0*k[0][1]+2.0*k[0][2]+k[0][3])/6.0;
        if (RADIUS > MAX_R) MAX_R =RADIUS;
        INCLINATION += (k[1][0]+2.0*k[1][1]+2.0*k[1][2]+k[1][3])/6.0;
        ANGLE_U   +=   (k[2][0]+2.0*k[2][1]+2.0*k[2][2]+k[2][3])/6.0;

        VELOCITY  +=   (k[3][0]+2.0*k[3][1]+2.0*k[3][2]+k[3][3])/6.0;
    }

}



#endif // RUNGEKUTTA_H
```

Файл "Modeling.h"

```
#include "Initialize.h"
#include "RungeKutta.h"
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "Modeling.h"
#include <QMessageBox>
#include <math.h>

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    db = QSqlDatabase::addDatabase("QODBC");
    db.setDatabaseName("DRIVER={Microsoft Access Driver (*.mdb,
*.accdb)};FIL={MS Access};DSN='';DBQ=.//database/engine.mdb");


    qDebug() << db.drivers();
    qDebug() << db.open();
    if(!db.open())
    {
```

```
        QMessageBox::critical(this, "Ошибка открытия Базы Данных",
db.lastError().text());

    }

    model = new QSqlTableModel(this,db);
    model->setTable("engine");
    model->select();
    ui->tableView->setModel(model);



    ui->widget->setBackground(QColor(20, 20, 20));
    ui->widget->xAxis->setBasePen(QPen(QColor(64,224,208), 1));
    ui->widget->yAxis->setBasePen(QPen(QColor(64,224,208), 1));
    ui->widget->xAxis->setTickLabelColor(QColor(64,224,208));
    ui->widget->yAxis->setTickLabelColor(QColor(64,224,208));
    ui->widget->xAxis->setBasePen(QPen(QColor(64,224,208), 1));
    ui->widget->yAxis->setBasePen(QPen(QColor(64,224,208), 1));
    ui->widget->xAxis->setTickPen(QPen(QColor(64,224,208), 1));
    ui->widget->yAxis->setTickPen(QPen(QColor(64,224,208), 1));
    ui->widget->xAxis->setSubTickPen(QPen(QColor(64,224,208), 1));
    ui->widget->yAxis->setSubTickPen(QPen(QColor(64,224,208), 1));
```

```
    ui->widget_4->setBackground(QColor(20, 20, 20));
    ui->widget_4->xAxis->setBasePen(QPen(QColor(64,224,208), 1));
    ui->widget_4->yAxis->setBasePen(QPen(QColor(64,224,208), 1));
    ui->widget_4->xAxis->setTickLabelColor(QColor(64,224,208));
    ui->widget_4->yAxis->setTickLabelColor(QColor(64,224,208));
    ui->widget_4->xAxis->setBasePen(QPen(QColor(64,224,208), 1));
    ui->widget_4->yAxis->setBasePen(QPen(QColor(64,224,208), 1));
    ui->widget_4->xAxis->setTickPen(QPen(QColor(64,224,208), 1));
    ui->widget_4->yAxis->setTickPen(QPen(QColor(64,224,208), 1));
    ui->widget_4->xAxis->setSubTickPen(QPen(QColor(64,224,208), 1));
    ui->widget_4->yAxis->setSubTickPen(QPen(QColor(64,224,208), 1));









}


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

```
void MainWindow::on_pushButton_2_clicked()
{
   QMessageBox::about(this, "О программе", "Самарский Университетет\n Кафедра космического машиностроения\n Разработчик: Хайруллин И.И.\n https://github.com/razbiralochka");
}


void MainWindow::on_pushButton_clicked()
{

      Fly_Time=ui->lineEdit->text().toDouble()*86400;
      Start_Orbit_Height=ui->lineEdit_4->text().toDouble();
      Start_Orbit_Inclination=ui->lineEdit_6->text().toDouble()*M_PI/180;
      Finally_Orbit_Height=ui->lineEdit_5->text().toDouble();
      Finally_Orbit_Inclination=ui->lineEdit_7->text().toDouble()*M_PI/180;
      Start_SC_Mass=ui->lineEdit_2->text().toDouble();
      Realitive_Construct_Mass=ui->lineEdit_12->text().toDouble();
      SSS_Realitive_Mass=ui->lineEdit_11->text().toDouble();
      Gas_Flow_Speed=ui->lineEdit_8->text().toDouble();
      Engine_Specific_Mass=ui->lineEdit_9->text().toDouble();
      Electro_Specific_Mass=ui->lineEdit_10->text().toDouble();
      EFFICIENCY=ui->lineEdit_3->text().toDouble();
```

```
    if (ui->radioButton->isChecked())
{

    SdelatPizdato();
}



QSqlQuery query;

    if (ui->radioButton_2->isChecked())
{

        query.exec("SELECT * FROM engine WHERE Код = " +
SelectedEngine);
        while (query.next())
        {
          name = query.value(1).toString();
          Engines_Thrust = query.value(2).toDouble() * 0.001;
          Gas_Flow_Speed = query.value(3).toDouble() * 1000.0;
          Engines_Power =  query.value(4).toDouble() * 1000.0;
```

```
            EFFICIENCY = query.value(5).toDouble() * 0.01;
          }

          SdelatZaebumba();
          ui->lineEdit_23->setText(name);




    }


    double MAXIMUM_PAYLOAD_MASS=0;

    if (ui->radioButton_3->isChecked())
{
        for (int i = 1; i<=18; i++)
        {
        query.exec("SELECT * FROM engine WHERE Код = " +
QString::number(i));
        while (query.next())
        {
          name = query.value(1).toString();
          Engines_Thrust = query.value(2).toDouble() * 0.001;
          Gas_Flow_Speed = query.value(3).toDouble() * 1000.0;
```

```
  Engines_Power =  query.value(4).toDouble() * 1000.0;
  EFFICIENCY = query.value(5).toDouble() * 0.01;
 }

 SdelatZaebumba();
 if (Payload_Mass > MAXIMUM_PAYLOAD_MASS)
 {
    MAXIMUM_PAYLOAD_MASS = Payload_Mass;
    EngineNumber=i;
    ui->lineEdit_23->setText(name);
 }
 }


 query.exec("SELECT * FROM engine WHERE Код = " +
QString::number(EngineNumber));
 while (query.next())
 {
  name = query.value(1).toString();
  Engines_Thrust = query.value(2).toDouble() * 0.001;
  Gas_Flow_Speed = query.value(3).toDouble() * 1000.0;
  Engines_Power =  query.value(4).toDouble() * 1000.0;
  EFFICIENCY = query.value(5).toDouble() * 0.01;
 }
```

```
    SdelatZaebumba();
    ui->lineEdit_23->setText(name);

}

    foo = (Gas_Mass)/(3*M_PI);
    Tank_Radius=100*round(pow(foo,1/3));
    Tank_CTR = round(Tank_Radius*1.5);
    Body_lenght=round(500*pow(Construct_Mass,1/3));
    SP_Square =0.5* Engines_Power/(1.3*0.29*0.866);
    Payload_R=round(sqrt(Payload_Mass/(0.02*1.5*M_PI)));
    EnginesCount=round(Engines_Thrust);
    ui->lineEdit_3->setText(QString::number(EFFICIENCY));
    ui->lineEdit_8->setText(QString::number(Gas_Flow_Speed));
    ui->lineEdit_18->setText(QString::number(Electro_Mass));
    ui->lineEdit_22->setText(QString::number(Payload_Mass));
    ui->lineEdit_21->setText(QString::number(SSS_Mass));
    ui->lineEdit_19->setText(QString::number(Engines_Mass));
    ui->lineEdit_20->setText(QString::number(Construct_Mass));
    ui->lineEdit_17->setText(QString::number(Engines_Power/1000));
    ui->lineEdit_16->setText(QString::number(Engines_Thrust/9.81));
    ui->lineEdit_15->setText(QString::number(Gas_Mass));
    ui->lineEdit_14->setText(QString::number(Delta_velocity/1000));
    ui->lineEdit_13->setText(QString::number(Initial_Speed/1000));
```

```
RK_STEPS =Fly_Time/100;
INITIAL_ACCELERATION = Engines_Thrust/Start_SC_Mass;
ANGLE_U = 0;
RADIUS = Start_Orbit_Radius;
VELOCITY=0;
INCLINATION=Start_Orbit_Inclination;
TIME = 0;

QVector <double> RADIUS_ARRAY(0);
QVector <double> INCLINATION_ARRAY(0);
QVector <double> TIME_ARRAY(0);

Runge_Kutta(&TIME_ARRAY,&RADIUS_ARRAY,&INCLINATION_ARRAY
);

ui->widget->xAxis->setRange(0,Fly_Time/86000);
```

```
ui->widget->yAxis->setRange(180*Start_Orbit_Inclination/M_PI,180*Finally_Orbit_Inclination/M_PI);
ui->widget->addGraph();
ui->widget->graph(0)->setPen(QPen(QColor(64,224,208), 2));
ui->widget->graph(0)->addData(TIME_ARRAY,INCLINATION_ARRAY);
ui->widget->replot();
ui->widget_4->xAxis->setRange(0,Fly_Time/86000);
ui->widget_4->yAxis->setRange(Start_Orbit_Radius/1000.0,MAX_R/1000.0);
ui->widget_4->addGraph();
ui->widget_4->graph(0)->setPen(QPen(QColor(64,224,208), 2));
ui->widget_4->graph(0)->addData(TIME_ARRAY,RADIUS_ARRAY);
ui->widget_4->replot();

RADIUS_ARRAY.clear();
INCLINATION_ARRAY.clear();
TIME_ARRAY.clear();

ui->label_31->setText("Зависимость наклонения орбиты (град) от времени (сут). Конечное наклонение: " + QString::number(180*INCLINATION/M_PI) + " град");
ui->label_32->setText("Зависимость радиуса орбиты (км) от времени (сут). Конечный радиус: " + QString::number(RADIUS/1000)+ " км");
```

```
    ExcelExport();
    QMessageBox::about(this, "Excel Export", "Данные занесены в таблицу data.csv");
}


void MainWindow::on_tableView_activated(const QModelIndex &index)
{
   SelectedEngine = ui->tableView->model()->data(index).toString();
   ui->lineEdit_23->setText(name);
}
```