Санкт-Петербургский государственный университет
Факультет прикладной математики – процессов управления
Кафедра управления медико-биологическими системами

Миронов Даниил Алексеевич

Выпускная квалификационная работа бакалавра

Моделирование динамики космического аппарата в поле звездной системы

Направление 01.03.02
Прикладная математика и информатика

Научный руководитель,
кандидат физ.-мат. наук,
доцент
Гончарова М. В.

Санкт-Петербург
2018

Содержание

Список обозначений

a	скалярная величина
$\mathbf{a}$	вектор
$\lvert\mathbf{a}\rvert$	модуль трехмерного вектора $\mathbf{a}$
$\mathbf{a} \cdot \mathbf{b}$	скалярное произведение векторов
$[\mathbf{a}, \mathbf{b}]$	векторное произведение векторов
$\dot{x}$	производная по времени
КА	космический аппарат
ka	индекс, соответствующий КА
$\odot$	индекс, соответствующий звезде
$\mathbf{r_{ka}}(t)$	координата КА в момент времени t
$\mathbf{v_{ka}}(t)$	скорость КА в момент времени t
$\mathbf{r^u_{ka}}(t)$	координата КА при управлении u
$\mathbf{v^u_{ka}}(t)$	скорость КА при управлении u

Введение

Полёты в космос и, в частности, к другим планетам издавна привлекали человека. С момента запуска первого искуственного спутника Земли 4 октября 1957 года и первой межпланетной автоматической станции в сторону венеры 12 февраля 1961 года и до сего дня прогресс в космических полётах не останавливается. Использование космических аппаратов–спутников Земли стало неотъемлимой частью жизни людей, с каждым годом отправляются новые аппараты к другим планетам.

Численное моделирование в целом является очень мощным инструментом в задачах небесной механики. Сложно переоценить его значение, потому как сфера его применения очень широка: данные, полученные в результате компьютерного моделирования, позволяют с высокой точностью прогрозировать явления астрономической природы на Земле, предсказывать движение небесных тел в будущем, выяснять их местонахождение в прошлом, проектировать космические миссии. Причем развитие современной техники позволяет проводить вычисления практически на любом компьютере, совсем не обязательно иметь суперкомпьютер под рукой. Это открывает возможности по проведению моделирования и принятия решений об управлении космическим аппаратом прямо на борту этого же космического аппарата.

В данной работе рассматривается подход к решению задачи о выборе управления космическим аппаратом путём моделировании динамики КА в поле звездной системы в общем случае, то есть при разработке программы параметры звездной системы заранее неизвестны. Таким образом, предполагается, что полученная компьютерная программа может использоваться на борту космического аппарата для принятия решений об управлении в случае полёта в неизвестную заранее обстановку или в случае непредвиденного изменения обстановки при условии наличия механизма получения данных о звездной системе и вычисления её параметров.

Несомненным плюсом такого подхода является его универсальность, алгоритмы решения задачи при его использовании не будут зависеть от параметров системы. Однако, на сегодняшний день, полёты практически всегда совершаются в уже известных системах, и в таком случае специализированный подход является более привлекательным. Поэтому использование

программ, аналогичных представленной в данной работе, предполагается в качестве второй или аварийной системы принятия решений о управлении.

Постановка задачи

Общая цель работы — проведение моделирования динамики космического аппарата в поле звездной системы, разделяется на следующие части: моделирование динамики тел в поле звездной системы, обеспечение механизма подачи управления на КА, поиск необходимого управления по задаваемым требованиям на состояние КА в момент времени. В связи с чем возникают следующие подзадачи:

1) Вывод системы дифференциальных уравнений, описывающих динамику тел без управления

2) Модификация полученной системы для обеспечения механизма подачи управления

3) Выбор и конкретизация численного метода для решения полученной системы уравнений

4) Определение алгоритмов поиска управления для различных классов требований

5) Реализация моделирования и поиска управления в виде компьютерной программы

Обзор литературы

При написании данной работы была использованна учебно–методическая литература общего характера, учебно– методическая литература по небесной механике и космической навигации, а также труды по численному решению задач небесной механики.

Пособие [4] использовалось как источник общих сведений о динамике, пособие [8] в качестве источника сведений о численных методах, из [6] был взят метод Зейделя, а пособие [1] выступало как руководство по использованию численных методов.

Вывод системы дифференциальных уравнений, описывающей динамику задачи N тел был произведен на основе книг [10] и [7]. Также изложенный в них материал позволил вывести приемлемые допущения для решения задачи.

Труд [3] использовался как руководство по выбору численного метода решения полученной системы, в нём представленно сравнение различных методов решения систем дифференциальных уравнений, используемых в задачах небесной механики. Также из [3] частично был взят метод Эверхарта, более подробное описание этого метода содержится в [2].

В качестве источника базовых сведений о подходах к управлению космическим аппаратом выступала книга [9]. Как источник более специализированных методов управления и сведений о космической навигации в целом использовалась книга [5].

Проектирование и разработка компьютерной программы производилась следуя принципам, изложенным в [12]. Книга [11] использовалась как руководство по языку java. Наконец, на основе [13] выполнена графическая составляющая программы, использующая пакет Canvas платформы javaFX.

1. Постановка основной задачи моделирования

В этом параграфе определим основные уравнения, на которых будет строиться дальнейшее моделирование.

Обозначим условия формализации задачи моделирования.

Межпланетное пространство считается абсолютной пустотой, то есть сопротивление среды отсутствует. Считается, что планеты не сталкиваются и сферы их гравитационного влияния не пересекаются, а КА не опускается ниже орбитальной высоты на планеты с атмосферой. Также из этого следует, что единственные тела в моделируемой звездной системе это звезда, планеты и КА, а единственные силы, действующие на каждое тело это гравитационные силы, подчиняющиеся закону тяготения Ньютона, порождаемые остальными телами. Все тела считаются материальными точками, за исключением того, что планеты и звезда имеют радиус — сближение планет ближе, чем на сумму их радиусов останавливает моделирование. Вращением планет вокруг собственной оси пренебрегается. Ориентацией КА в пространстве также пренебрегается — считается, что работа маневровых двигателей не влияет на положение центра масс КА в пространстве, или же используются гиросиловые стабилизаторы; также считается, что всегда есть достаточно времени, для изменения ориентации КА в пространстве. Эффекты ОТО не учитываются.

Таким образом, при условии наличия начальных данных, получаем прямую задачу динамики для каждого тела, то есть положение каждого тела описывается дифференциальным уравнением:

$$\ddot{\mathbf{r}} m = \mathbf{F},$$

где $\mathbf{F}$ — равнодействующая сила, $\mathbf{r}$ — радиус–вектор тела, а m — его масса. Состояние системы описывает система дифференциальных уравнений:

$$\ddot{\mathbf{r_i}} m_i = \mathbf{F_i}, \quad i = \overline{1, N} \tag{1}$$

где $\mathbf{F_i}$ — равнодействующая сила, $\mathbf{r_i}$ — радиус–вектор, а m_i — масса i–го тела, N — число тел[4].

Вспомним формулировку классической задачи N тел: в пустоте находится N материальных точек, массы которых известны и равняются m_i. Пусть попарное взаимодействие точек подчинено закону тяготения Ньютона, и пусть силы гравитации аддитивны. Пусть известны начальные на момент времени $t = 0$ положения и скорости каждой точки $r_i|_{t=0} = r_{i0}$, $v_i|_{t=0} = v_{i0}$. Требуется найти положения точек для всех последующих моментов времени[10]. Таким образом, полученная прямая задача динамики это и есть задача N тел. Тогда уравнение (1) можно переписать как:

$$\ddot{\mathbf{r}}_\mathbf{i} m_i = \sum_{j \neq i}^{N} \mathbf{F_{ij}}, \quad i = \overline{1, N}$$

или

$$\begin{cases} \dot{\mathbf{r}}_\mathbf{i} = \mathbf{v_i}, \\ \dot{\mathbf{v}}_\mathbf{i} = \sum\limits_{j \neq i}^{N} \mathbf{F_{ij}}, \end{cases} \qquad i = \overline{1, N} \tag{2}$$

где $\mathbf{F_{ij}} = G m_i \frac{\mathbf{r_j} - \mathbf{r_i}}{|\mathbf{r_j} - \mathbf{r_i}|^3}$; $m_i, \mathbf{r_i}, \mathbf{v_i}$, — масса, радиус–вектор и скорость i-го тела соответственно, а G — гравитационная постоянная[10].

Теперь специализируем формулировку задачи под наш случай: среди рядовых тел (планет) есть два особых: звезда и КА. Считается, что звезда всегда находится в начале координат ($\mathbf{r}_\odot = \mathbf{0}$), а также считается возможным предельный переход $m_i / M_\odot \to 0$. Считается, что на КА можно подать тягу $\mathbf{u}$, также считается возможным предельный переход $m_{ka}/m_i \to 0$.

В таком случае $\mathbf{F_{kaj}} = G m_i \dfrac{\mathbf{r_j} - \mathbf{r_i}}{|\mathbf{r_j} - \mathbf{r_i}|^3} + \dfrac{\mathbf{u}}{m_{ka}}$, и если принять для всех остальных тел нулевую силу $\mathbf{F_{t_i}}$, то систему (2) можно переписать как:

$$\begin{cases} \dot{\mathbf{r}}_\mathbf{i} = \mathbf{v_i}, \\ \dot{\mathbf{v}}_\mathbf{i} = \sum\limits_{j \neq i}^{N} G m_i \dfrac{\mathbf{r_j} - \mathbf{r_i}}{|\mathbf{r_j} - \mathbf{r_i}|^3} + \dfrac{\mathbf{F_{t_i}}}{m_i}, \end{cases} \qquad \mathbf{r_0} = \mathbf{r}(t_0), \dot{\mathbf{r}}_\mathbf{0} = \mathbf{v_0} = \dot{\mathbf{r}}(t_0), \tag{3}$$

где $m_i, \mathbf{r_i}, \mathbf{v_i}$ — масса, радиус–вектор и скорость i-го тела соответственно, i изменяется от 1 до N, G — гравитационная постоянная, а $\mathbf{F_{t_i}}$ — сила тяги двигателя i–го тела, $\mathbf{F_{t_i}} = \mathbf{0}$, для всех i, кроме ka, для которого $\mathbf{F_{t_{ka}}} = \dfrac{\mathbf{u}}{\mathbf{m_{ka}}}$. Тогда система (3) — основное уравнение (задача), решение которого будет моделироваться.

2. Численное решение поставленной задачи

На данный момент задача N тел для $N > 3$ не может быть решена аналитически (ряды Зундмана не имеют практической ценности, в связи с чем не учитываются) [7]. Поэтому для аналитического решения задачи перехода между состояниями КА используют метод сфер гравитационного действия, в которых считается, что на аппарат действует только тело, в сфере которого он находится, и таким образом переходят к задаче двух тел [5]. Сфера гравитационного действия — сфера, внутри которой $f'/g' < f''/g''$, где g' — ускорение тела при движении вокруг планеты, f' — возмущение в этом ускорении от звезды, g'' — ускорение тела при движении вокруг звезды, f'' — возмущение в этом ускорении от планеты.

Другой путь решения задачи — использование численных методов для решения задачи N тел в общем виде и моделирование полученного результата. Данная работа была посвящена этому методу.

Под решением подразумевается реккурентная формула для приближенного решения задачи (3): $\mathbf{r_n} = \mathbf{r_{n-1}} + \mathbf{S_{r_n}}, \mathbf{v_n} = \mathbf{v_{n-1}} + \mathbf{S_{v_n}}$, где $\mathbf{S_{r_n}}, \mathbf{S_{v_n}}$ есть результат работы выбранного метода. Если такая формула получена, то считается, что нам известно состояние системы $\boldsymbol{r}$ и $\boldsymbol{v}$ в любой момент времени $t > t_0$, или на любом шаге n. Также стоит упомянуть, что для сохранения условия $\mathbf{r}_\odot = \mathbf{0}$ вышеупомянутая формула преобразуется в $\mathbf{r_n} = \mathbf{r_{n-1}} + \mathbf{S_{r_n}} - \mathbf{r}_{\odot_\mathbf{n}}$. Для получения решения задачи будем использовать метод Эверхарта, однако в процессе работы были рассмотрены также метод разложения в ряд Тейлора и классический метод Рунге–Кутты. Подробнее о каждом из них далее.

2.1. Разложение в ряд Тейлора

Метод разложения в ряд Тейлора — один из старейших методов решения дифференциальных уравнений. Главный его недостаток — для достижения приемлемой точности необходимо вычислять производные правой части уравнения, что есть весьма трудоемкое занятие. В связи с этим данный метод использовался только в качестве проверки адекватности настроек и параметров программной реализации решения задачи, как самый простой (в случае, когда производные не вычислялись, об этом ниже).

Рассмотрим одно тело. Имея начальные данные
$\mathbf{r_0} = \mathbf{r}(t_0), \dot{\mathbf{r}}_\mathbf{0} = \mathbf{v_0} = \dot{\mathbf{r}}(t_0)$, необходимо найти $\mathbf{r}(t_0 + h)$, где h — шаг по времени. Тогда разложим $\mathbf{r}(t_0 + h)$ в ряд Тейлора:

$$\mathbf{r}(t_0 + h) = \mathbf{r_0} + \sum_{i=1}^{\infty} \frac{\mathbf{r}^{(i)}(t_0)}{i!} h^i, \tag{4}$$

а если взять только первые два члена суммы, и распространить на все шаги, то из (4) получим

$$\mathbf{r_n} \approx \mathbf{r_{n-1}} + h\mathbf{v_{n-1}} + \frac{h^2}{2}\dot{\mathbf{v}}(t_{n-1}),$$

где $\dot{\mathbf{v}}(t_0)$ очевидным образом получается из (3) [8]. Как нетрудно видеть, в таком случае не требуется вычислять производные правой части. Полученное приближение является весьма грубым, однако требует очень малое количество вычислений, и моделирование с применением такого метода работает очень быстро, что позволяет легко обнаруживать критические ошибки программирования.

2.2. Классический метод Рунге–Кутты

В общем случае задача N тел является жесткой задачей [3], и использование явных методов Рунге–Кутты является некорректным. Однако, в случае орбит, близких к круговым, жесткость задачи нивелируется, и использование явных методов становится возможным. Явные методы требуют меньше вычислений, чем неявные, и моделирование на явном методе работает быстрее. В нашем случае классический метод Рунге–Кутты использовался на начальных данных для круговых орбит для проверки корректности программной реализации поиска управления на КА, для которой метод разложения в ряд Тейлора уже неприемлем, но точности, обеспечиваемой этим методом ещё хватает. Полное обоснование и вывод метода можно посмотреть в [8]. Запишем ещё раз систему (3) в виде, удобном для

применения метода Рунге–Кутты:

$$\begin{cases} \dot{\mathbf{r_i}} = \mathbf{v_i} = \mathbf{g_i}(t, \mathbf{r}, \mathbf{v}), \\ \dot{\mathbf{v}}_\mathbf{i} = \sum\limits_{j \neq i}^{N} G m_i \frac{\mathbf{r_j} - \mathbf{r_i}}{|\mathbf{r_j} - \mathbf{r_i}|^3} + \mathbf{F_{t_i}} = \mathbf{f_i}(t, \mathbf{r}, \mathbf{v}), \end{cases} \tag{5}$$

Тогда расчетная схема принимает вид:

$$\mathbf{r_{n+1}} = \mathbf{r_n} + \frac{h}{6}(\mathbf{q_1} + \mathbf{2q_2} + \mathbf{2q_3} + \mathbf{q_4}),$$

$$\mathbf{v_{n+1}} = \mathbf{v_n} + \frac{h}{6}(\mathbf{k_1} + \mathbf{2k_2} + \mathbf{2k_3} + \mathbf{k_4}),$$

$$\mathbf{k_1} = \mathbf{f}(t_n, \mathbf{r_n}, \mathbf{v_n}),$$

$$\mathbf{q_1} = \mathbf{g}(t_n, \mathbf{r_n}, \mathbf{v_n}),$$

$$\mathbf{k_2} = \mathbf{f}(t_n + \frac{h}{2}, \mathbf{r_n} + \frac{h}{2}\mathbf{q_1}, \mathbf{v_n} + \frac{h}{2}\mathbf{k_1}),$$

$$\mathbf{q_2} = \mathbf{g}(t_n + \frac{h}{2}, \mathbf{r_n} + \frac{h}{2}\mathbf{q_1}, \mathbf{v_n} + \frac{h}{2}\mathbf{k_1}),$$

$$\mathbf{k_3} = \mathbf{f}(t_n + \frac{h}{2}, \mathbf{r_n} + \frac{h}{2}\mathbf{q_2}, \mathbf{v_n} + \frac{h}{2}\mathbf{k_2}),$$

$$\mathbf{q_3} = \mathbf{g}(t_n + \frac{h}{2}, \mathbf{r_n} + \frac{h}{2}\mathbf{q_2}, \mathbf{v_n} + \frac{h}{2}\mathbf{k_2}),$$

$$\mathbf{k_4} = \mathbf{f}(t_n + h, \mathbf{r_n} + h\mathbf{q_2}, \mathbf{v_n} + h\mathbf{k_2}),$$

$$\mathbf{q_4} = \mathbf{g}(t_n + h, \mathbf{r_n} + h\mathbf{q_2}, \mathbf{v_n} + h\mathbf{k_2}),$$

$$\mathbf{r}(t_0) = \mathbf{r_0},$$

$$\mathbf{v}(t_0) = \mathbf{v_0},$$

где h — шаг по времени t. Этот метод имеет четвертый порядок точности.

2.3. Метод Эверхарта

В 1973 г. Э. Эверхарт предложил интегратор, разработанный им специально для численного исследования орбит, и продемонстрировал его высокую эффективность в задачах кометной динамики. Этот метод остается одним из самых популярных в решении задач небесной механики, его

мы и будем использовать для финальной версии моделирования. Полное обоснование и вывод метода можно посмотреть в [2], [3].

Использовался четырех–стадийный метод, разбиение Гаусса–Лежандра, т.е. коэффициенты c_1, c_2, c_3, c_4 это решения уравнения $\frac{d^4}{dt^4}(t^4(t-1)^4)$,
а именно:

$$c_1 = \frac{1}{2}(1 - \sqrt{\frac{1}{35}(15 - 2\sqrt{30})}),$$

$$c_2 = \frac{1}{2}(1 + \sqrt{\frac{1}{35}(15 - 2\sqrt{30})}),$$

$$c_3 = \frac{1}{2}(1 - \sqrt{\frac{1}{35}(15 + 2\sqrt{30})}),$$

$$c_4 = \frac{1}{2}(1 + \sqrt{\frac{1}{35}(15 + 2\sqrt{30})}),$$

или $c_1 \approx 0.330009, c_2 \approx 0.669991, c_3 \approx 0.0694318, c_4 \approx 0.930568$. Такое разбиение и число стадий даёт восьмой порядок метода. Тогда рассчетная схема принимает вид (h — шаг по времени):

$$\mathbf{y_1} = \mathbf{v_{n-1}} + h\sum_{j=1}^{4}\frac{\mathbf{a_j}}{j}c_1^j$$

$$\mathbf{y_2} = \mathbf{v_{n-1}} + h\sum_{j=1}^{4}\frac{\mathbf{a_j}}{j}c_2^j$$

$$\mathbf{y_3} = \mathbf{v_{n-1}} + h\sum_{j=1}^{4}\frac{\mathbf{a_j}}{j}c_3^j$$

$$\mathbf{y_4} = \mathbf{v_{n-1}} + h\sum_{j=1}^{4}\frac{\mathbf{a_j}}{j}c_4^j \qquad (6)$$

$$\mathbf{z_1} = \mathbf{r_{n-1}} + h\sum_{j=1}^{4}\frac{\mathbf{b_j}}{j}c_1^j$$

$$\mathbf{z_2} = \mathbf{r_{n-1}} + h\sum_{j=1}^{4}\frac{\mathbf{b_j}}{j}c_2^j$$

$$\mathbf{z_3} = \mathbf{r_{n-1}} + h\sum_{j=1}^{4} \frac{\mathbf{b_j}}{j} c_3^j$$

$$\mathbf{z_4} = \mathbf{r_{n-1}} + h\sum_{j=1}^{4} \frac{\mathbf{b_j}}{j} c_4^j \tag{7}$$

$$\mathbf{k_1} = \mathbf{f}(t_{n-1} + hc_1, \mathbf{y_1}, \mathbf{z_1})$$

$$\mathbf{k_2} = \mathbf{f}(t_{n-1} + hc_2, \mathbf{y_2}, \mathbf{z_2})$$

$$\mathbf{k_3} = \mathbf{f}(t_{n-1} + hc_3, \mathbf{y_3}, \mathbf{z_3})$$

$$\mathbf{k_4} = \mathbf{f}(t_{n-1} + hc_4, \mathbf{y_4}, \mathbf{z_4})$$

$$\mathbf{q_1} = \mathbf{g}(t_{n-1} + hc_1, \mathbf{y_1}, \mathbf{z_1})$$

$$\mathbf{q_2} = \mathbf{g}(t_{n-1} + hc_2, \mathbf{y_2}, \mathbf{z_2})$$

$$\mathbf{q_3} = \mathbf{g}(t_{n-1} + hc_3, \mathbf{y_3}, \mathbf{z_3})$$

$$\mathbf{q_4} = \mathbf{g}(t_{n-1} + hc_4, \mathbf{y_4}, \mathbf{z_4})$$

$$\boldsymbol{\alpha_1} = \boldsymbol{k_1}$$

$$\boldsymbol{\alpha_2} = \frac{\boldsymbol{k_2} - \boldsymbol{\alpha_1}}{c_2 - c_1}$$

$$\boldsymbol{\alpha_3} = \frac{\frac{\boldsymbol{k_3} - \boldsymbol{\alpha_1}}{c_3 - c_1} - \boldsymbol{\alpha_2}}{c_3 - c_2}$$

$$\boldsymbol{\alpha_4} = \frac{(c_4 - c_3)(\frac{\boldsymbol{k_4} - \boldsymbol{\alpha_1}}{c_4 - c_2} - \boldsymbol{\alpha_2})}{c_4 - c_2} - \frac{\boldsymbol{\alpha_3}}{c_4 - c_3}$$

$$\boldsymbol{\beta_1} = \boldsymbol{q_1}$$

$$\boldsymbol{\beta_2} = \frac{\boldsymbol{q_2} - \boldsymbol{\beta_1}}{c_2 - c_1}$$

$$\boldsymbol{\beta_3} = \frac{\frac{\boldsymbol{q_3} - \boldsymbol{\beta_1}}{c_3 - c_1} - \boldsymbol{\beta_2}}{c_3 - c_2}$$

$$\boldsymbol{\beta_4} = \frac{(c_4 - c_3)(\frac{\boldsymbol{q_4} - \boldsymbol{\beta_1}}{c_4 - c_2} - \boldsymbol{\beta_2})}{c_4 - c_2} - \frac{\boldsymbol{\beta_3}}{c_4 - c_3}$$

$$d_{ii} = 1, d_{i0} = 0, d_{ij} = d_{i-1,j-1} - c_{i-1} d_{i-1,j}, (i > j > 0)$$

$$\boldsymbol{a_1} = \sum_{i=1}^{4} d_{i1} \boldsymbol{\alpha_i}$$

$$\boldsymbol{a_2} = \sum_{i=2}^{4} d_{i2}\boldsymbol{\alpha_i}$$

$$\boldsymbol{a_3} = \boldsymbol{\alpha_3} + d_{43}\boldsymbol{\alpha_4}$$

$$\boldsymbol{a_4} = \boldsymbol{\alpha_4}$$

$$\boldsymbol{b_1} = \sum_{i=1}^{4} d_{i1}\boldsymbol{\beta_i}$$

$$\boldsymbol{b_2} = \sum_{i=2}^{4} d_{i2}\boldsymbol{\beta_i}$$

$$\boldsymbol{b_3} = \boldsymbol{\beta_3} + d_{43}\boldsymbol{\beta_4}$$

$$\boldsymbol{b_4} = \boldsymbol{\beta_4}$$

где функции $\boldsymbol{f}$ и $\boldsymbol{g}$ берем из (5). Тогда уравнения (6) и (7) представляют собой неявные уравнения от $\boldsymbol{y}$ и $\boldsymbol{z}$ ($\boldsymbol{y} = \mathbf{y}(\mathbf{a}(\boldsymbol{\alpha}(\mathbf{k}(\mathbf{y},\mathbf{z})))), \mathbf{z} = \mathbf{z}(\mathbf{b}(\boldsymbol{\beta}(\mathbf{q}(\mathbf{y},\mathbf{z})))))$, и решаются методом Зейделя [6]. Как только величины $\boldsymbol{y}$ и $\boldsymbol{z}$ получены, по ним вычисляются $\boldsymbol{a}$ и $\boldsymbol{b}$, и приближенное решение на шаге n будет:

$$\mathbf{v_n} = \mathbf{v_{n-1}} + h\sum_{j=1}^{4}\frac{\boldsymbol{a_j}}{j},$$

$$\mathbf{r_n} = \mathbf{v_{r-1}} + h\sum_{j=1}^{4}\frac{\boldsymbol{b_j}}{j}, \tag{8}$$

Таким образом, требуемое решение получено.

3. Задание управления на космический аппарат

Поиск и задание упраления $\mathbf{u}$ на КА должны решить некоторую поставленную задачу о требуемом состоянии КА в нужный момент времени, то есть в общем случае это требование $\mathbf{r}^{\mathbf{u}}_{\mathbf{ka}}(t^*) = \mathbf{r}^*$ и (или) $\mathbf{v}^{\mathbf{u}}_{\mathbf{ka}}(t^*) = \mathbf{v}^*$, а решением задачи будет являтся такое управление $\mathbf{u}$, что выполняется указанное равенство. В общем случае подобную задачу мы рассматривать не будем, потому что чаще всего требования относятся к какому-либо классу — например, требование о пересечении с каким-либо телом, изменение высоты апоцентра и так далее. Классы требований, рассмотренные в этой работе и соответствующие алгоритмы поиска управления приведены далее.

3.1. Формализация орбит

Динамика тел в поле звездной системы имеет цикличную природу, поэтому для четкого определения задачи на поиск управления на КА из общих представлений о желаемом, необходимо иметь формализацию циклов движения тел — орбит. Здесь, как и законах Кеплера считается возможным введение эллиптической орбиты, пренебрегая отклонениями, то есть считать в пределах сферы гравитационного влияния возможной замену задачи N тел на задачу двух тел, однако моделирование проводится все равно по численному решению задачи N тел.

Классическая формализация в случае орбитального движения в поле звезды или планеты — кеплеровы элементы орбиты [4], [5]. Вспомним их, это:

большая полуось (a)
экстрентриситет (e)
наклонение (i)
долгота восходящего узла (Ω)
аргумент перицентра (ω),

Первые два определяют форму орбиты, третий, четвёртый и пятый — ориентацию плоскости орбиты по отношению к базовой системе координат. Шестой элемент определяет положение тела на орбите, и здесь не указан,

по тому как в данной работе состояние тела рассматривается отдельно от его орбиты. В таком виде формализация подходит к задаче двух тел, либо когда все тела вращаются вокруг одного. В нашем случае в формализацию также входит радиус–вектор тела — центра орбиты $\mathbf{r}(t)$. К тому же, для моделирования была выбрана трехмерная декартова система координат, которая хотя и не является самым лучшим вариантом для рассмотрения задач небесной механики, очень удобна для применения в программировании, так как для отображения состояния системы на экране, координаты из любой системы все равно нужно переводить в декартову. Тогда требуется переформулировать вышеуказанные данные о орбите в приемлемую для нас форму. Оказывается достаточным хранить три вектора: нормаль плоскости орбиты $\mathbf{n}$, координаты перицентра $\mathbf{r_{peri}}$ и апоцентра $\mathbf{r_{apo}}$ — это полностью определяет положение орбиты в пространстве. Покажем, что из этих трех векторов можно получить все кеплеровы элементы:

$$a = \frac{|\boldsymbol{r_{apo}}| + |\boldsymbol{r_{peri}}|}{2},$$

$$e = \frac{|\boldsymbol{r_{apo}}| - |\boldsymbol{r_{peri}}|}{|\boldsymbol{r_{apo}}| + |\boldsymbol{r_{peri}}|},$$

$$i = \arccos\left(\mathbf{n} \cdot (0,0,1)\right),$$

вектор пересечения с плоскостью Oxy: $\mathbf{p} = [\mathbf{n}, (0,0,1)]$,

направление на точку весеннего равноденствия: $\mathbf{ov}$,

$$\Omega = \arccos\left(\frac{\mathbf{ov} \cdot \mathbf{p}}{|\mathbf{ov}|\ |\mathbf{p}|}\right),$$

$$\omega = \arccos\left(\frac{\mathbf{p} \cdot \mathbf{r_{peri}}}{|\mathbf{p}|\ |\mathbf{r_{peri}}|}\right).$$

Следует упомянуть, что для полного описания орбиты требуются в том числе и данные о теле, которое является центром этой орбиты. Однако этот аспект отлагается до межпланетных маневров, а до этого времени считается что КА находится все время в сфере гравитационного влияния одного тела, которое также находится в начале координат. На этом формализацию орбит можно считать законченной.

3.2. Требования на скорость космического аппарата в точке орбиты

Первое действие в любом маневре — это изменение скорости КА в точке уже имеющейся стартовой орбиты. Иными словами, это требование $\mathbf{r}^{\mathbf{u}}_{\mathbf{ka}}(t^*) = \mathbf{r}^*$ и $\mathbf{v}^{\mathbf{u}}_{\mathbf{ka}}(t^*) = \mathbf{v}^*$, где $\mathbf{r}^*$ это точка орбиты, или $\mathbf{r_{ka}}(t^*)$ при нулевом управлении $\mathbf{u} = \mathbf{0}$. Надо сказать, что в таком виде подобную задачу можно решить только при неограниченном управлении, или очень малой разницей между $\mathbf{v_{ka}}(t^*)$ при нулевом управлении и $\mathbf{v}^*$ которую обозначим $\Delta\mathbf{v}$, т.е. когда время в которое $\mathbf{u} \neq \mathbf{0}$ стремится к нулю. Однако в реальном случае управление ограниченно, а $\Delta\mathbf{v}$ может быть вовсе не так мало. Тогда ограничимся требованием $\mathbf{v}^{\mathbf{u}}_{\mathbf{ka}}(t^*) = \mathbf{v}^*$, а контроль за отклонением от планируемой точке оставим на следующие этапы маневра. Также в ходе работы экспериментально было выяснено, что это отклонение будет невелико. Например, при маневре с $\Delta\mathbf{v}$ в 300 м/с с текущей скоростью порядка 30 км/с отклонение будет около 3 км, при условии, что двигатель КА способен обеспечивать ускорение в 5 м/с^2.

Таким образом, требуется компенсировать $\Delta\mathbf{v} = \mathbf{v}^* - \mathbf{v^0}$, где $\mathbf{v^0}$ есть скорость КА в требуемое время t^* при нулевом управлении $\mathbf{u} = \mathbf{0}$. То есть получим, что из $\mathbf{v^0} + \boldsymbol{\Delta}\mathbf{v} = \mathbf{v}^*$, $\Delta\mathbf{v}$ — это дополнительная скорость, которую необходимо приобрести КА. Обозначим максимальный возможный модуль управления как U, $|\mathbf{u}| \leqslant U$. Так как управление в нашем случае это тяга двигателя, то U обуславливается особенностями конструкции двигателя КА. Также, если вспомнить допущения, введенные в самом начале, а именно: «Ориентация КА в пространстве также не учитывается, считается, что работа маневровых двигателей не влияет на положение центра масс КА в пространстве, или же используются гиросиловые стабилизаторы, также считается, что всегда есть достаточно времени, для изменения ориентации КА в пространстве». Таким образом ограничение на направление управления $\mathbf{u}/|\mathbf{u}|$ отсутствует. Из-за дискретности времени в моделировании, будем считать, что t^* достигается на шаге n. Также получаем, что если шаг по времени есть h, согласно (3), очевидно, что дополнительная скорость, получаемая за один шаг это $\mathbf{u}h/m_{ka}$, соответственно максимально возможное приращение скорости есть Uh/m_{ka}.

Отметим, что $\Delta\mathbf{v}$ используется как первое приближение необходимой

дополнительной скорости, алгоритм поиска финальной дополнительной скорости будет приведён далее, а сейчас опишем механизм преобразования подаваемой дополнительной скорости $\Delta\mathbf{v}_*$ в правило подачи управления на КА.

Требуется рассмотреть два случая, которые возникают при сравнении величин $|\Delta\mathbf{v}_*|$ и Uh/m_{ka}. Рассмотрим сначала случай $|\Delta\mathbf{v}_*| \leqslant Uh/m_{ka}$. В этом случае на шаге $n-1$ подаётся управление $\mathbf{u} = \Delta\mathbf{v}_* m_{ka}/h$, и на шаге n, $\mathbf{u}$ опять зануляется. В случае же $|\Delta\mathbf{v}_*| \geqslant Uh/m_{ka}$ вычислим целое s по правилу $sU < |\Delta\mathbf{v}_*| < (s+1)U$, иначе говоря поделим $|\Delta\mathbf{v}_*|$ нацело на U. Обозначим $o_s = |\Delta\mathbf{v}_*| - sU$. Тогда на $n-s-1$ шаге подаётся управление $\mathbf{u} = (\Delta\mathbf{v}_*/|\Delta\mathbf{v}_*|)o_s m_{ka}/h$, s шагов, начиная с $n-s$ подаётся $\mathbf{u} = (\Delta\mathbf{v}_*/|\Delta\mathbf{v}_*|)Um_{ka}/h$, и на n–ом шаге $\mathbf{u}$ зануляется. Обозначим этот механизм как (9).

Теперь опишем алгоритм поиска нужной дополнительной скорости $\Delta\mathbf{v}^*$. Для этого применяется одомерная минимизация методом золотого сечения[1]. Одномерность достигается за счет того, что напраление дополнительной скорости всегда берется $\Delta\mathbf{v}/|\Delta\mathbf{v}|$, и поиск идет только на модуль. Минимизируемая функция задаётся следующим образом:

$$f(x) = |\mathbf{v}^* - \mathbf{v}^{\mathbf{x}}|^2,$$

где $\mathbf{v}^{\mathbf{x}}$ это скорость КА на шаге n при поданной дополнительной скорости $(\Delta\mathbf{v}/|\Delta\mathbf{v}|)x$. Минимизация ведется на отрезке $\left[-10^6|\Delta\mathbf{v}|, 10^6|\Delta\mathbf{v}|\right]$. Выбор такого большого отрезка обусловелен тем, что размер отрезка мало влияет на количество итераций метода минимизации, а минимум у функции f один — когда КА будет находиться в требуемом состоянии. Обозначим $\Phi = (1+\sqrt{5})/2$, отрезок на текущей итерации $[a,b]$, точку деления с прошлой итерации x_-, $f(x_-) = f_-$. Тогда каждая итерация выгляди так:

1) Если $|a-b| < |b| \cdot 0{,}001$ возвращается $\frac{a+b}{2}$, остановка минимизации.

2) Рассчитываются точки деления: $x_1 = b - \frac{b-a}{\Phi}, x_2 = a + \frac{b-a}{\Phi}$.
   Если x_- не определен — $y_1 = f(x_1)$, $y_2 = f(x_2)$,
   если $x_1 = x_-$, то $y_1 = f_-$,
   если $x_2 = x_-$, то $y_2 = f_-$.

3) Если $y_1 \geqslant y_2 : a = x_1,\ x_- = x_2,\ f_- = y_2,$
если $y_1 < y_2 : b = x_2,\ x_- = x_1,\ f_- = y_1,$
возврат к шагу 1)

В итоге, при требовании $\mathbf{v}^{\mathbf{u}}_{\mathbf{ka}}(t^*) = \mathbf{v}^*$, необходимо использовать механизм подачи управления (9), на вход которому подать дополнительную скорость $(\Delta\mathbf{v}/|\Delta\mathbf{v}|)x$, где $\Delta\mathbf{v} = \mathbf{v}^* - \mathbf{v}^{\mathbf{0}}$, где $\mathbf{v}^{\mathbf{0}}$ есть скорость КА в требуемое время t^* при нулевом управлении $\mathbf{u} = \mathbf{0}$, а x получен в результате работы минимизирующего алгоритма, представленного выше.

3.3. Требования на параметры орбиты

Требование на параметры орбиты в общем случае, это требование вида $\mathbf{r}^{\mathbf{u}}_{\mathbf{apo}} = \mathbf{r}^*_{\mathbf{apo}}, \mathbf{r}^{\mathbf{u}}_{\mathbf{peri}} = \mathbf{r}^*_{\mathbf{peri}}, \mathbf{n}^{\mathbf{u}} = \mathbf{n}^*$. По сути, это несколько требований вида $\mathbf{r}^{\mathbf{u}}_{\mathbf{ka}}(t^*) = \mathbf{r}^*$, а именно требование, что найдутся такие t_1^*, t_2^*, что $\mathbf{r}^{\mathbf{u}}_{\mathbf{ka}}(t_1^*) = \mathbf{r}^*_{\mathbf{apo}}, \mathbf{r}^{\mathbf{u}}_{\mathbf{ka}}(t_2^*) = \mathbf{r}^*_{\mathbf{peri}}$, а для всех остальных t выполняется неравенство $|\mathbf{r}^*_{\mathbf{peri}}| \leqslant |\mathbf{r}^{\mathbf{u}}_{\mathbf{ka}}(t)| \leqslant |\mathbf{r}^*_{\mathbf{apo}}|$ и $\mathbf{r}^{\mathbf{u}}_{\mathbf{ka}}(t) \cdot \mathbf{n}^* = 0$. В данном случае имеется в виду, что $t > t_m$, где t_m это время, после которого управление $\mathbf{u} = \mathbf{0}$. Здесь и далее, единичное выполнение механизма (9) будем называть выполнением простого маневра. Таким образом вышесказанное можно переформулировать, как $t > t_m$, где t_m это время окончания выполнения всех простых маневров. Требования на параметры орбиты тоже разбиваются на некоторые классы требований, которые мы рассмотрим далее.

Также стоит сказать, что решение задачи по удовлетворению таких требований имеет два подхода: через определение и постановку требований на скорость из пункта 3.2, и без этого. В самом алгоритме разница небольшая, смысл в разделении этих подходов — разница в программировании. Первый вариант предпочтительнее в случае наличия мощной вычислительной техники, так как предлагает более структурированный код программы, использование уже написанных методов и сохранение уровней абстракции, то есть принципов, описанных в [12]. Второй вариант обратно, вынуждает писать больше кода и нарушает абстракцию, однако, требует меньше вычислений, в следствии чего программа работает быстрее. В распоряжении автора не находилось достаточно мощной выислительной техники, в связи с чем был выбран второй вариант. Однако, далее будут приведены описания

обоих подходов.

Выполнение требований на параметры орбиты требует исполнения простого маневра, или исполнения последовательности простых маневров. В случае последовательности, задача разбивается на последовательное выполнение задач с исполнением одного простого маневра, поэтому сначала будем рассматривать задачи, требующие одного простого маневра. Последовательность простых маневров будем называть сложным маневром. Определим простой маневр более четко. Простой маневр — алгоритм подачи управления на КА, принимающий на вход направление маневра $\mathbf{e}, |\mathbf{e}| = \mathbf{1}$, время окончания t и модуль дополнительной скорости x. Тогда сам алгоритм есть механизм (9), которому на вход подается $\mathbf{e}x$, а время t достигается на шаге n. Время исполнения маневра есть либо h, либо $(s+1)h$ в соответствии с двумя случаями, определёнными в (9).

Направление маневра $\mathbf{e}$ для решения задачи необходимо определить аналитически для каждого класса задач и задать заранее. Время завершения исполнения маневра t_b необходимо также задать заранее, t_b достигается на шаге b. Тогда опишем механизм определения модуля дополнительной скорости для маневра, выполняющего задачу.

Сначала опишем механизм, использующий первый подход. В этом случае используется минимизация функции

$$f(x) = |\mathbf{r}^*_{\mathbf{apo}} - \mathbf{r}^{\mathbf{x}}_{\mathbf{apo}}|^{\mathbf{2}} + |\mathbf{r}^*_{\mathbf{peri}} - \mathbf{r}^{\mathbf{x}}_{\mathbf{peri}}|^{\mathbf{2}} + |\mathbf{n}^* - \mathbf{n}|^{\mathbf{2}},$$

где $\mathbf{r}^{\mathbf{x}}_{\mathbf{apo}}$, $\mathbf{r}^{\mathbf{x}}_{\mathbf{peri}}$, $\mathbf{n}$ — параметры орбиты после исполнения требования вида $\mathbf{v_{ka}}(t_b) = \mathbf{e}x$. Надо отметить, что вид функции f здесь приведен для общего случая и может быть изменен в зависимости от класса задачи и от вида требований. Минимизация производится методом золотого сечения, описанным выше на отрезке $\left[-10^6|\mathbf{v^0_{ka}}(t_b)|, 10^6|\mathbf{v^0_{ka}}(t_b)|\right]$, где $\mathbf{v^0_{ka}}(t_b)$ это скорость КА в момент t_b без подачи управления. На выходе минимизации получаем x^*, тогда маневр, решающий задачу есть маневр, решающий задачу удовлетворения требования $\mathbf{v^u_{ka}}(t_b) = \mathbf{e}x^*$. Механизм, использующий второй подход схож с первым, отличие заключается в том, что в определении функции $\mathbf{r}^{\mathbf{x}}_{\mathbf{apo}}$, $\mathbf{r}^{\mathbf{x}}_{\mathbf{peri}}$, $\mathbf{n}$ —параметры орбиты после выполнения маневра, которму на вход подано $\mathbf{e}$, x, t_b, и на выход механизма подается не требование на скорость, а маневр, на вход которому подано $\mathbf{e}$, x^*, t_b. Получается,

что в первом подходе при каждом вычислении функции $f(x)$ запускается минимизация в вычислении решения для удовлетворения требования по скорости, а во втором такого не происходит, откуда и разница в количестве вычислений.Получаем, что теперь для каждого класса задач необходимо определить направление $\mathbf{e}x$ и время окончания маневров t_b.

Рассмотрим класс требований на апоцентр и перицентр по отдельности, то есть вида $\mathbf{r^u_{apo}} = \mathbf{r^*_{apo}}$ и $\mathbf{r^u_{peri}} = \mathbf{r^*_{peri}}$. Удовлетворение этих требований будет производиться при помощи траектории Гомана–Цандера [4], [9], согласно которой направление маневра будет $\mathbf{v_{ka}}(t)$, в случае, когда КА движется по круговой орбите. В случае требования вида $|\mathbf{r^u_{apo}}| = apo \geqslant |\mathbf{r^0_{peri}}|$ время t_b будет временем прохождения КА перицентра текущей орбиты при нулевом управлении: $\mathbf{r^0_{ka}}(t_b) = \mathbf{r_{peri}}$, из соображений, что в момент прохождения перицентра можно представить, что КА был на круговой орбите высотой $|\mathbf{r_{peri}}|$, часть маневра уже совершил, и осталось довершить этот воображаемый маневр с круговой орбиты. Минимизируемая функция выглядит в этом случае как $f(x) = |apo - |\mathbf{r^x_{apo}}||^2$. Аналогично, при требовании $|\mathbf{r^u_{peri}}| = peri \leqslant |\mathbf{r^0_{apo}}|$ время t_b будет временем прохождения апоцентра, $f(x) = |peri - |\mathbf{r^x_{peri}}||^2$. В случае круговой стартовой орбиты и отсутствия апоцентра и перицентра t_b можно взять любое.

Тогда маневр выхода на круговую орбиту представляет собой маневр, удовлетворяющий требованию $|\mathbf{r^u_{apo}}| = apo$, где $apo = |\mathbf{r_{peri}}|$ — высота текущего перицентра, либо наоборот, $|\mathbf{r^u_{peri}}| = peri$, где $peri = |\mathbf{r_{apo}}|$ — высота текущего апоцентра. Получаем, что удовлетворение требований вида $\mathbf{r^u_{apo}} = \mathbf{r^*_{apo}}$ и $\mathbf{r^u_{peri}} = \mathbf{r^*_{peri}}$ в общем случае представляет собой сложный маневр, включающий в себя сначала выход на круговую орбиту, а затем удовлетворение требования на высоту апоцентра или перицентра, в которых t_b будет временем прохождения через точку пересечения круговой орбиты с прямой, определяемой вектором $\mathbf{r^*_{apo}}$ или $\mathbf{r^*_{peri}}$ соответственно. Подробнее сложные маневры будут описаны далее, а теперь рассмотрим требование вида $\mathbf{n^u} = \mathbf{n}^*$.

Время t_b — это время прохождения КА через точку пересечения текущей орбиты и прямой, по которой пересекаются текущая плоскость орбиты, и требуемая. Иными словами, это момент времени, в который КА оказывается в требуемой плоскости $\mathbf{r^0_{ka}} \cdot \mathbf{n}^* = 0$. Задача изменения плоскости

решается немного иначе, чем задачи изменения апоцентра и перицентра, для неё всегда на выходе получаем требование на скорость, по причине того, что ничего не нужно минимизировать с использованием этого требования. Если скорость в требовании удовлетворяет уравнению $\mathbf{v}^* \cdot \mathbf{n}^* = 0$, то его исполнение исполнит требование на плоскость. Модуль новой скорости остается тот же, что и без управления $|\mathbf{v}^{\mathbf{0}}_{\mathbf{ka}}(t_b)| = |\mathbf{v}^*|$, а направление находится снова при помощи минимизации методом золотого сечения. Минимизируемая функция выглядит следующим образом: $f(x) = |\mathbf{n}^* \cdot (\mathbf{v}^{\mathbf{0}}_{\mathbf{ka}}(t_b) + x\mathbf{n}^{\mathbf{0}})|$, где $\mathbf{n}^{\mathbf{0}}$ это нормаль плоскости без управления, или, нормаль стартовой плоскости. Минимизация производится на отрезке $[-10^3, 10^3]$. Тогда при полученном результате минимизации x^* итоговое требование на скорость будет $\mathbf{v}^{\mathbf{u}}_{\mathbf{ka}}(t_b) = \mathbf{v}^{\mathbf{0}}_{\mathbf{ka}}(t_b) + x^*\mathbf{n}^{\mathbf{0}}$.

Следует упомянуть о значении равенств в требованиях и выполнимости маневров. Все равенства в требованиях, такие как $a^u = a^*$ и $\mathbf{b}^{\mathbf{u}} = \mathbf{b}^*$, из-за использования минимизации и особенностей хранения данных при программировании в целом, на самом деле, представляют собой неравнества вида $|a^u - a^*| < 0{,}01|a^*|$ и $|\mathbf{b}^{\mathbf{u}} - \mathbf{b}^*| < 0{,}01|\mathbf{b}^*|$ соответственно. Подобная замена равенства на неравенство происходит каждый раз при проверке двух величин на равенство, при назначении одной величины равной другой равенство сохраняется. Также можно заметить, что во всех случая минимизации предполагается, что минимум функции $f(x)$ — это ноль. В случае, если минимум недостаточно близок к нулю, маневр и соответствующее требование считаются неисполнимыми.

3.4. Сложные маневры

Сложный маневр — это, как было сказано выше, последовательность простых маневров. Сложный маневр считается неисполнимым, если хотя бы один из простых маневров в его составе неисполним. Задание сложного маневра — суть задание последовательности простых маневров и их параметров, а так же задание временного параметра первого простого маневра. Если временной параметр не задан, считается что он равен любому времени $t > t_0$, где t_0 — время начала моделирования. Далее опишем сложные маневры для некоторых классов задач.

Выход на орбиту с указанным апоцентром или перицентром уже был

описан выше, однако повторим его в этом разделе. Выход на требуемую орбиту $\mathbf{r^u_{apo}} = \mathbf{r^*_{apo}}$ или $\mathbf{r^u_{peri}} = \mathbf{r^*_{peri}}$ подразумевает под собой снача выход на круговую орбиту, а затем удовлетворение требования на высоту апоцентра или перицентра, в которых t_b будет временем прохождения через точку пересечения круговой орбиты с прямой, определяемой вектором $\mathbf{r^*_{apo}}$ или $\mathbf{r^*_{peri}}$ соответственно. Выход на круговую орбиту производится в перицентре в случае требования на апоцентр, и в апоцентре, в случае требования на перицентр. Здесь подразумевается, что $\mathbf{n^0} = \mathbf{n}^*$. Если это не так, то сложный маневр суть выход на круговую орбиту, затем смена плоскости на требуемую, затем удовлетворение требования на высоту. Тогда удовлетворение требования на параметры орбиты в общем случае, $\mathbf{r^u_{apo}} = \mathbf{r^*_{apo}}, \mathbf{r^u_{peri}} = \mathbf{r^*_{peri}}, \mathbf{n^u} = \mathbf{n}^*$, суть выход на круговую орбиту в перицентре с высотой $|\mathbf{r^0_{peri}}|$, затем переход на орбиту в плоскости с нормалью $\mathbf{n}^*$, затем удовлетворение требования на перицентр с концом маневра в точке пересечения текущей круговой орбиты и прямой, проходящей через перицентр и апоцентр, затем удовлетворение требования на высоту апоцентра. В случае, когда $|\mathbf{r^*_{peri}}| > |\mathbf{r^0_{peri}}|$, в ходе маневра по изменению перицентра условием $peri \leqslant |\mathbf{r^0_{apo}}|$ следует пренебречь.

Как уже упоминалось в пункте 3.2, при удовлетворении требования на скорость, иначе говоря исполнении простого маневра, во время t_b происходит отклонение $\mathbf{r^*_{ka}}(t_b)$ от $\mathbf{r^0_{ka}}(t_b)$. Обычно это отклонение невелико и им можно пренебречь, однако при большой дополнительной скорости оно может увеличиваться, и если по расчетам требуется чтобы орбита проходила через точку $\mathbf{r^0_{ka}}(t_b)$, можно добавить к простому маневру поправочный, превратив его таким образом в сложный. Для выхода на круговую орбиту с большей точностью нужно выполнить маневр выхода на круговую орбиту ещё раз: во второй раз дополнительная скорость не будет столь большой, и отклонением можно будет пренебречь. Для изменения высоты апоцентра или перицентра с большей точностью нужно выполнить сложный маневр с требованием на апоцентр и перицентр, где вторым параметром будет текущий перицентр или апоцентр соответственно. Для ещё большей точности в вышеуказанных маневрах можно добавить ещё одну итерацию, и, вообще говоря, сколько угодно, но обычно это нецелесообразно.

Далее описывается удовлетворение требования на состояние КА в

общем случае. Это требование вида $\mathbf{r^u_{ka}}(t^*) = \mathbf{r}^*$ и $\mathbf{v^u_{ka}}(t^*) = \mathbf{v}^*$. Для определения сложного маневра, удовлетворяющего таким требованиям сначала определяется маневр, удовлетворяющий требованиям $\mathbf{r^u_{ka}}(t) = \mathbf{r}^*$ и $\mathbf{v^u_{ka}}(t) = \mathbf{v}^*$, где t — любое с учетом $t > t_0$, и рассчитывается $\Delta t = t - t_b$, где t_b это время начала первого простого маневра. Удовлетворение требований с любым временем производится следующим образом: если $|\mathbf{r}^*| \geqslant |\mathbf{r^0_{apo}}|$ производится маневр по требованию $\mathbf{r^u_{apo}} = \mathbf{r^*_{apo}} = \mathbf{r}^*$, и после него ставится требование на скорость $\mathbf{v^u_{ka}}(t^a) = \mathbf{v}^*$, где t^a это время прохождения точки $\mathbf{r}^*$ с нулевым управлением после маневра по смене апоцентра. Если $|\mathbf{r}^*| \leqslant |\mathbf{r^0_{peri}}|$, то производится аналогичный маневр, только вместо маневра по смене апоцентра производится маневр по смене перицентра с требованием $\mathbf{r^u_{peri}} = \mathbf{r^*_{peri}} = \mathbf{r}^*$. В случае $|\mathbf{r^0_{peri}}| \leqslant |\mathbf{r}^*| \leqslant |\mathbf{r^0_{apo}}|$ производится любой из двух маневров. Назовем этот маневр (10). После получения Δt рассчитывается $t_b^0 = t^* - \Delta t$ — первое приближение временного параметра первого простого маневра. Далее производится минимизация методом золотого сечения на отрезке $\left[-10^6 t_b^0, 10^6 t_b^0\right]$. Минимизируемая функция $f(t)$ задается как $f(t) = |\mathbf{r_{ka}}(t^*) - \mathbf{r}^*|^2 + |\mathbf{v_{ka}}(t^*) = \mathbf{v}^*|^2$, где $\mathbf{r_{ka}}(t^*)$ и $\mathbf{v_{ka}}(t^*)$ это координата и скорость КА в момент времени t^* после выполнения (10) с параметром по времени t. Найденная точка минимума — t_b^*. Если $f(t_b^*)$ недостаточно близка к нулю, то требования считаются неисполнимыми, в противном случае искомый маневр — маневр (10) с параметром по времени t^*.

3.5. Межпланетные маневры

До этого момента рассматривались только маневры с одним телом–центром орбиты. Далее будут рассматриваться сложные маневры с переходом из сферы гравитационного влияние одного тела в сферу влияния другого. Для этого необходимо обозначить условия использования простых и сложных маневров из предыдущего раздела. Получается, что в случае, когда тело i является центром орбиты КА, рассматриваемые ранее $\mathbf{r_{ka}}$ и $\mathbf{v_{ka}}$ на самом деле были $\mathbf{r^{\tilde{i}}_{ka}}$ и $\mathbf{v^{\tilde{i}}_{ka}}$ — относительные координата и скорость соответственно, вычисляемые как $\mathbf{r^{\tilde{i}}_{ka}} = \mathbf{r_{ka}} - \mathbf{r_i}$, $\mathbf{v^{\tilde{i}}_{ka}} = \mathbf{v_{ka}} - \mathbf{v_i}$, где $\mathbf{r_{ka}}, \mathbf{r_i}, \mathbf{v_{ka}}, \mathbf{v_i}$ это координаты и скорости КА и i–ой планеты соотвественно в системе координат со звездой в начале координат $\mathbf{r}_\odot = \mathbf{0}$. Соответственно орбита в предыдущих пунктах это орбита в относительных координатах, и

все координаты и скорости КА в предыдущих пунктах тоже суть относительные координаты и скорости. Осуществляя таким образом связь между траекторией, координатами и скоростью из системы координат, связанной с центром орбиты и с системой координат, связанной со звездой, использование ранее описанных маневров считается возможным. Далее описываются некоторые классы межпланетных маневров.

Радиус сферы гравитационного действия тела i обозначается r_g^i, $r_g^\odot = \infty$. Тогда маневр выхода на орбиту звезды с орбиты любой планеты это маневр по изменению высоты апоцентра в котором $apo > r_g^i$.

В таком случае удовлетворение требования на состояние или орбиту КА вне сфер действия планет в общем случае это сначала маневр выхода на орбиту звезды, а затем маневр по требованию в смысле, описанном ранее. Способ выполнения задачи по переходу из сферы действия одной планеты в сферу другой планеты в общем случае это выход на орбиту звезды, и после этого постановка требования $|\mathbf{r}_{\mathbf{ka}}^{\tilde{\mathbf{i}}}(t)| < r_g^i$, где t — время прибытия в сферу назначения, а i — индекс планеты со сферой назначения. Надо отметить, что это наиболее общий подход, возможно не являющийся оптимальным в конкретном случае. Рассмотрим задачу перехода с орбиты одной планеты на орбиту другой. По сути это задача по переходу из сферы действия стартовой планеты в сферу действия планеты назначения, и после этого выполнение маневра по удовлетворению требования на параметры относительной орбиты.

Далее описывается более оптимальный метод решения задачи перехода между планетами. Общая идея взята из [5]. Для этого сначала определяется метод по решению задачи перехода между двумя фиксированными состояниями КА на орбите одного тела в фиксированное время, между $\mathbf{r}_{\mathbf{ka}}(t_1) = \mathbf{r}_\mathbf{1}^*$ и $\mathbf{r}_{\mathbf{ka}}(t_2) = \mathbf{r}_\mathbf{2}^*$, т.е. требуется определить скорость $\mathbf{v}_{\mathbf{ka}}(t_1) = \mathbf{v}^*$, которая обеспечит $\mathbf{r}_{\mathbf{ka}}(t_2) = \mathbf{r}_\mathbf{2}^*$. Сначала определяется плоскость орбиты перехода, нормаль которой будет $\mathbf{n}^* = [\mathbf{r}_\mathbf{1}^*, \mathbf{r}_\mathbf{2}^*]$. После этого определяется начальное приближение искомой скорости $\mathbf{v}^\mathbf{r}$, которая берется как скорость, необходимая для круговой орбиты, а именно $\mathbf{v}^\mathbf{r} = ([\mathbf{r}_\mathbf{1}^*, \mathbf{n}^*]/|[\mathbf{r}_\mathbf{1}^*, \mathbf{n}^*]|)\sqrt{GM/|\mathbf{r}_\mathbf{1}^*|}$, где G — гравитационная постоянная, а M — масса тела–центра орбиты. Искомая скорость определяется методом покоординатной минимизации [1] функции $f(x, y) = |\mathbf{r}_\mathbf{2}^* - \mathbf{r}_\mathbf{2}^{\mathbf{x},\mathbf{y}}|^2$, где $\mathbf{r}_\mathbf{2}^{\mathbf{x},\mathbf{y}}$ это координата КА во время

t_2, если $\mathbf{v_{ka}}(t_1) = x([\mathbf{r_1^*}, \mathbf{n}^*]/|[\mathbf{r_1^*}, \mathbf{n}^*]|) + y(\mathbf{r_1^*}/|\mathbf{r_1^*}|)$. Методом одномерной минимизации выступает метод золотого сечения. Минимизация по обеим координатам ведётся на отрезке $\left[-10^6\sqrt{GM/|\mathbf{r_1^*}|}, 10^6\sqrt{GM/|\mathbf{r_1^*}|}\right]$. Результаты процесса минимизации — x^*, y^*. Если $f(x^*, y^*)$ недостаточно близка к нулю, переход считается невозможным. В противном случае искомая скорость $\mathbf{v}^* = x^*([\mathbf{r_1^*}, \mathbf{n}^*]/|[\mathbf{r_1^*}, \mathbf{n}^*]|) + y^*(\mathbf{r_1^*}/|\mathbf{r_1^*}|)$.

Тогда задача перехода между планетами решается следующим способом: сначала определяется время старта и время прилёта t_1, t_2 (на самом деле получится, что t_2 это не совсем время прилета, а время прилета, если у планеты назначения нулевой радиус), что определяет положение стартовой планеты и планеты назначения: $\mathbf{r_i}(t_1), \mathbf{r_j}(t_2)$, где i, j — индексы планеты старта и планеты назначения соответственно. После чего решается задача перехода между состояниями $\mathbf{r_i}(t_1)$, $\mathbf{r_j}(t_2)$ в сфере влияния звезды так, как будто масса планеты старта нулевая. После чего находится точка пересечения полученной орбиты перехода и границы сферы гравитационного действия планеты старта $\mathbf{r_s}$ (радиус сферы действия рассчитывается с обычной массой планеты), определяется время, в которое КА при движении по орбите перехода попадает в эту точку t_s, определяется $\mathbf{v_{ka}}(t_s) = \mathbf{v_s}$; $\mathbf{r_{ka}}(t_s) = \mathbf{r_s}$. После чего для составления маневра сначала ставится требование на состояние КА в сфере действия стартовой планеты (точнее на её границе) $\tilde{\mathbf{r}}^{\mathbf{i}}_{\mathbf{ka}}(t_s) = \mathbf{r_s} - \mathbf{r_i}(t_s)$, $\tilde{\mathbf{v}}^{\mathbf{i}}_{\mathbf{ka}}(t_s) = \mathbf{v_s} - \mathbf{v_i}(t_s)$, а затем ставится требование $\tilde{\mathbf{v}}^{\mathbf{j}}_{\mathbf{ka}}(t^s) = \mathbf{v_p}$, где t^s — время соприкосновения КА с поверхностью планеты назначения, т.е. истинное время прилёта, а $\mathbf{v_p}$ это скорость поверхности планеты относительно её центра масс в точке соприкосновения В данной работе вращение планет вокруг своей оси не было включенно в моделирование, так что требование на совпадение скорости со скоростью поверхности здесь приведено для универсальности алгоритма.

На этом теоретическую часть можно считать законченной, и далее следует описание практической реализации приведенных выше выкладок — программы.

4. Практическая реализация моделирования

Практическая реализация — компьютерная программа, написана на языке java с использованием платформы javaFX [13] для графического отображения состояния системы. Выбранный язык достаточен для выполнения поставленной задачи, его функционал даёт достаточную свободу в проэктировании архитектуры программы. Также, по сравнению например с C++, является более удобным для реализации графического отображения системы и предоставляет более простой функционал для работы с памятью.

Использовалась декартова система координат, в которой плоскость OXY совпадает с плоскостью эклиптики. Для хранения трехмерных векторов был создан класс с тремя полями типа double, для хранения N–мерных векторов для численных методов решения системы (3) использовались связные списки ArrayList. Размерность для расстояния: $1 = 10^6$ км, для массы 10^{30} кг, шаг по времени — одна минута. Это значит, что например, если переменная double x= 0,14321 хранит расстояние, то её значение есть 143210 км.

Полный код программы доступен по ссылке:
`https://bitbucket.org/MironovDaniil/system-new/src/master/`

4.1. Архитектура программной реализации

Классы в программе разделены на три пакета — graphics, model и flightControl, отвечающие соответственно за графическую составляющую, моделирование и задание управения на КА. Схематично взаимодействие пакетов приведено на рисунке 1. Пакет model поставляет данные о состоянии моделируемой системы в graphics и flightControl. Пакет flightControl обрабатывает данные, полученные от model, в зависимости от выбранной задачи рассчитывает управление на КА, и возвращает его обратно в model. Пакет graphics представляет графически данные о состоянии системы, полученные от model. Также из него осуществляется запуск программы, установка начальных данных и постановка задачи на управление КА, то есть пакеты model и flightControl получают начальные данные из него.

Далее следует описание и схематичное представление пакетов и основных классов. Важно отметить, что не все классы и методы будут описаны

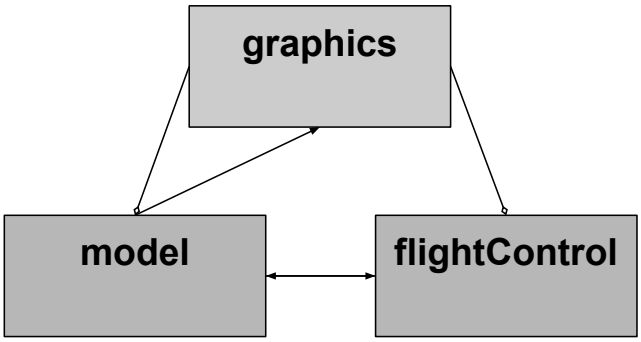

Рис. 1: Взаимодействие пакетов.

и отражены на схемах, а только отражающие соответствие программы и описанной выше теории.

Сначала описывается структура пакета model. Схематично она отражена на рисунке 2. ModelSit содержит как поле список из SpaceObject — абстрактного класса, наследниками которого являются Planet и Rocket, таким образом экземпляр ModelSit хранит информацию о текущем состоянии конкретной моделируемой системы. Vector3 описывает трехмерный вектор, Trajectory хранит траекторию и вычисляет параметры орбиты. Solver — класс, отражающий численный метод решения системы (3). На самом деле класса с таким названием в программе нет, как уже упомяналось в разделе 2, использовалось несколько методов, поэтому Solver символизирует выбранный в данный момент метод.

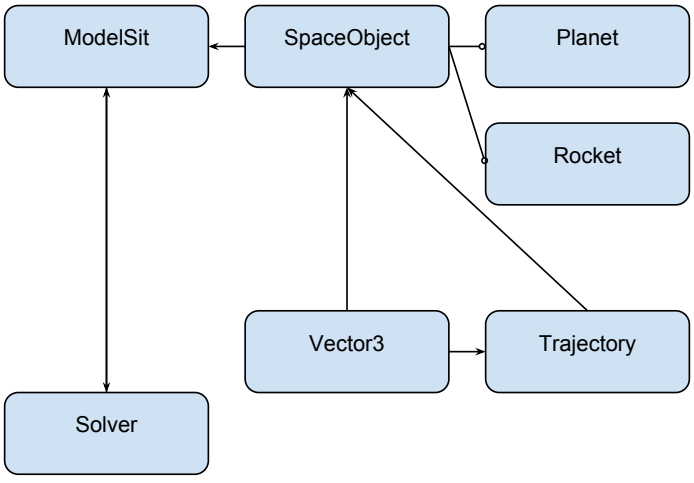

Рис. 2: Структура пакета model.

Vector3 — класс, описывающий трехмерный вектор. Он содержит методы, соответствующие операциям с векторами — сложение, скалярное и векторное произведение векторов, умножение на число и тому подобное. Схема класса приведена на рисунке 3.

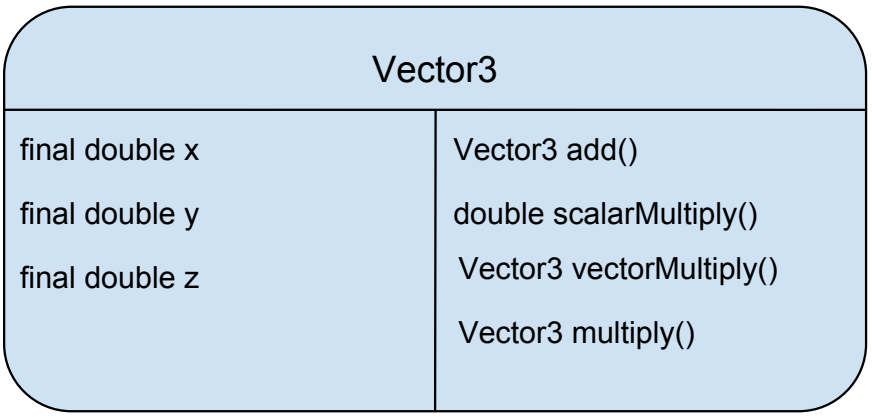

Рис. 3: Структура класса Vector3.

SpaceObject — абстрактный класс, описывающий любое тело в модели. Содержит координату, скорость и ускорение тела, а так же его траекторию. Позволяет устанавливать координату, скорость и ускорение, этим занимается класс ModelSit. Схема класса приведена на рисунке 4.

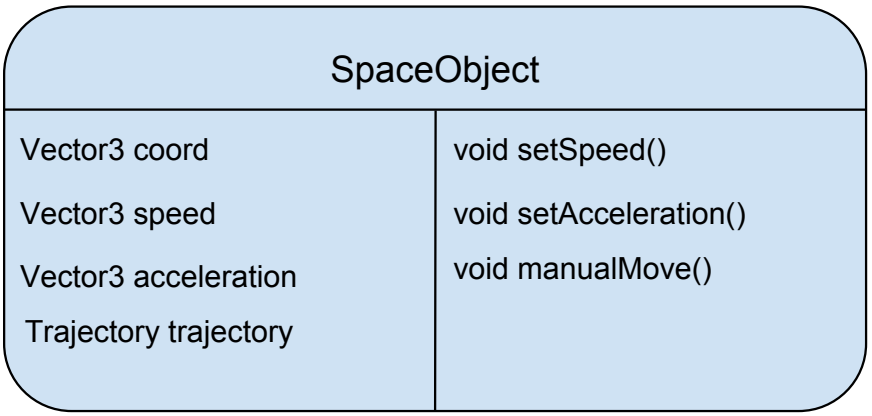

Рис. 4: Структура класса SpaceObject.

Planet это наследник SpaceObject, описывающий планеты и звезду. В дополнение к родительскому классу содержит радиус планеты, радиус сферы действия и массу звезды (в случае если экземпляр класса оказывается звездой, масса звезды записывается как null, и радиус сферы считается бесконечным). Никаких особенных методов не содержит. Схема класса приведена на рисунке 5.

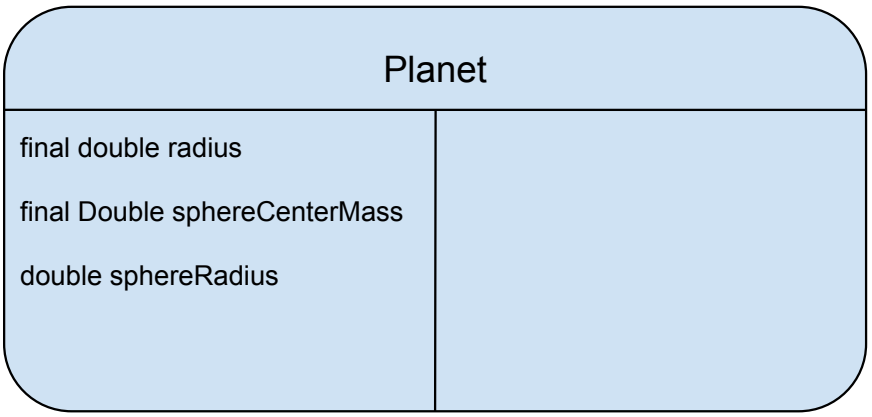

Рис. 5: Структура класса Planet.

Rocket также является наследником SpaceObject, он описывает КА. Дополнительно к родительскому классу содержит значение максимальной тяги, вектор текущей тяги, планету–центр текущей орбиты, и относительные координату, скорость и траекторию. Методы setOrbitCenter() и changeCentTrigger() соответственно пытается назначить планету центром орбиты, и проверяет принадлежность КА к сфере влияния планеты. Первый возвращает true в случае успешного назначения, второй в случае выхода КА из сферы действия планеты–аргумента. Назначение проходит успешно в том случае, когда КА находится в сфере действия планеты–аргумента. Метод setRelatives() рассчитывает относительные координату и скорость. setThrust() устанавливает текущую тягу. Схема класса приведена на рисунке 6.

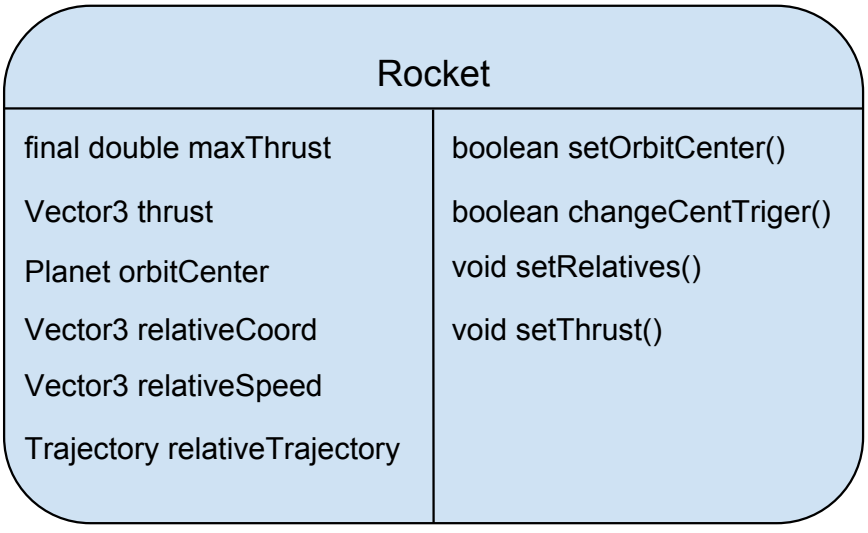

Рис. 6: Структура класса Rocket.

Trajectory описывает траекторию, или орбиту. Он хранит список то-

чек, составляющих траекторию, стабильна ли (в этом случае замкнута ли) орбита, апоцентр, перицентр, большую полуось, эксцентриситет, нормаль плоскости орбиты, время периода и сколько раз тело совершило оборот по этой орбите. Особенный метод в классе один — writeTraj, остальные просто возвращают поля класса. Метод writeTraj записывает новую точку в список, и производит рассчёт всех параметров орбиты. Схема класса приведена на рисунке 7.

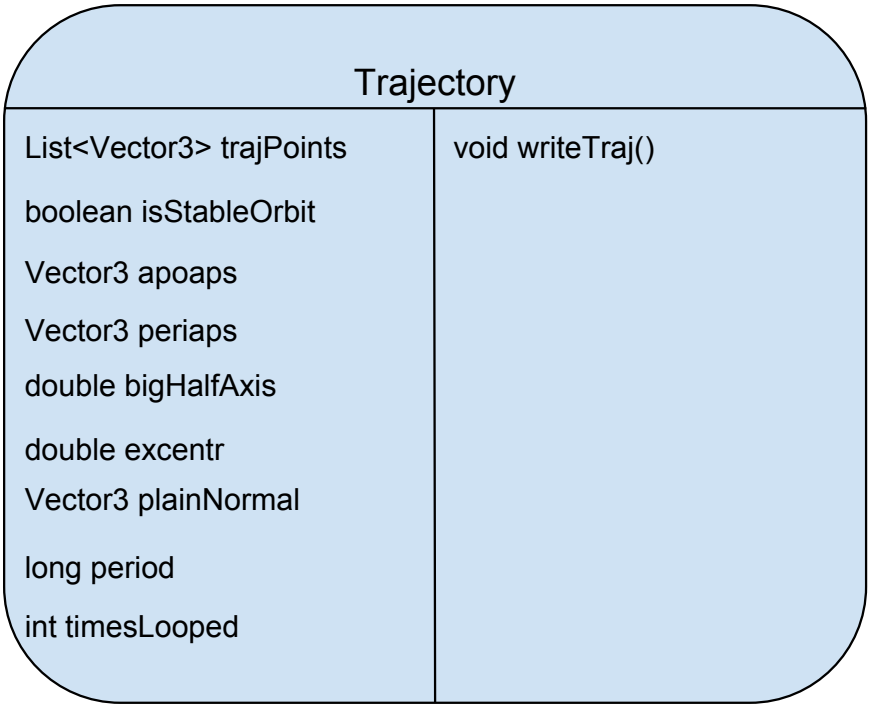

Рис. 7: Структура класса Trajectory.

Solver соответствует выбранному численному методу. Класс не содержит полей, но содержит два интерфейса, имеющих один метод run(), и представляющих собой функции из правой части системы (3), и статический класс–контейнер для результата. Единственный метод solve() численно решает систему (3), пакует результат в класс–контейнер и возвращает его. Схема класса приведена на рисунке 8.

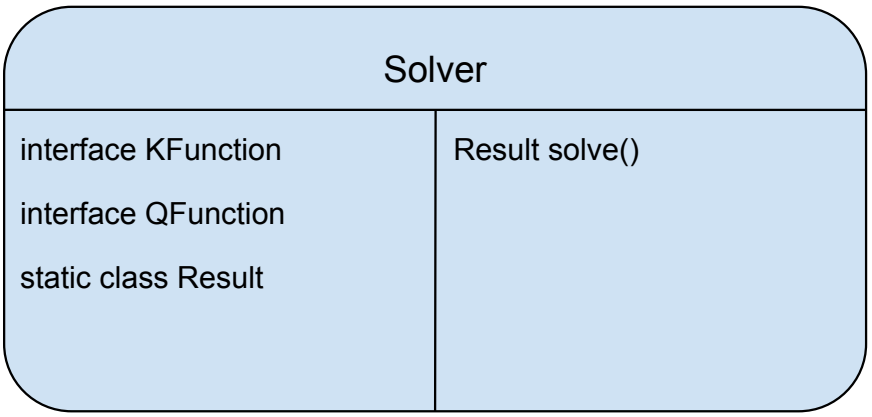

Рис. 8: Структура класса Solver.

ModelSit — основной класс пакета, в нём хранятся все данные о текущем состоянии системы, он содержит метод nextMinute, который двигает систему на минуту вперед, используя Solver. Схема класса приведена на рисунке 9.

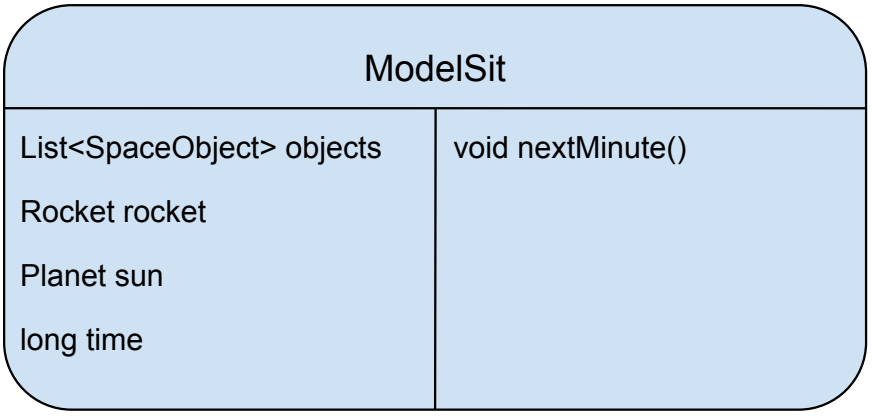

Рис. 9: Структура класса ModelSit.

Далее описывается структура пакета flightControl. Этот пакет соответствует пунктам 3, то есть осуществляет поиск и подачу управления на КА. Схематично структура пакета отражена на рисунке 10. ManeuverPlanner — основной класс, задающий управление в каждом конкретном случае. SingularNode, LongSimpleManeuver и NodeSeries это соответственно требование на скорость или простой маневр, простой маневр со сложной проверкой (из пункта 3.3) и сложный маневр. Для описания требований используется класс VesselState, хранящий состояние КА в конкретное время. GoldenSection отражает метод золотого сечения, используется в простых маневрах. CoordMin отражает метод покоординатной минимизации, используется в некоторых сложных маневрах.

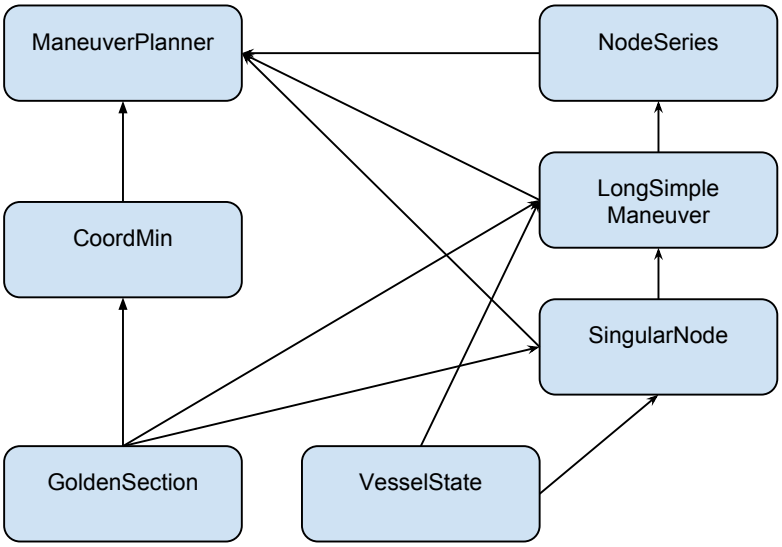

Рис. 10: Структура пакета flightControl.

VesselState — класс, хранящий состояние КА в момент времени, содержит в качестве полей векторы координаты и скорости, и значение времени. Никаких особых методов не имеет. Схема класса приведена на рисунке 11.

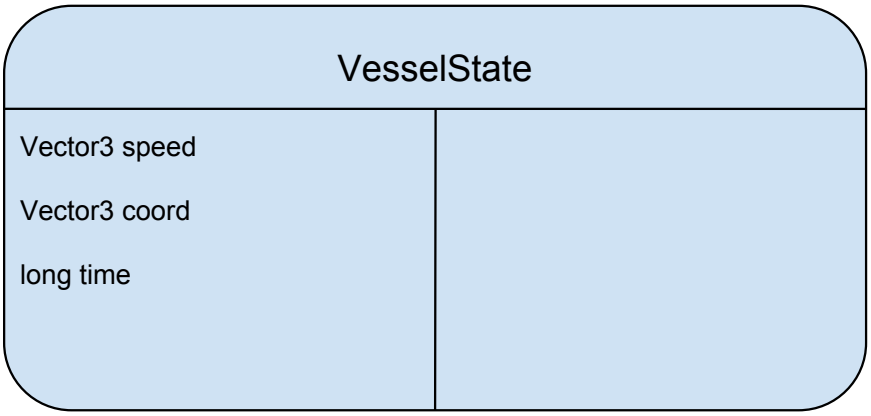

Рис. 11: Структура класса VesselState.

CoordMin и GoldenSection устроенны похоже на Solver, имеют один метод, составляющий суть метода, содержат интерфейс, отражающий минимизируемую функцию. CoordMin также содержит класс–контейнер для результата. Их схемы приведены на рисунках 12 и 13.

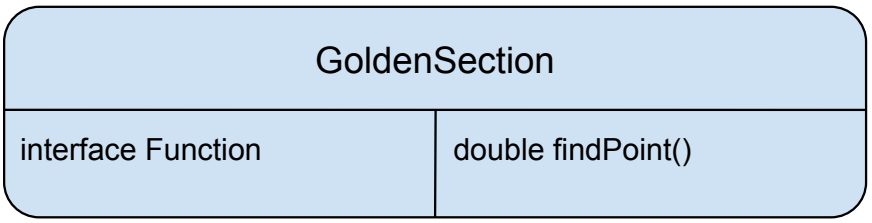

Рис. 12: Структура класса GoldenSection.

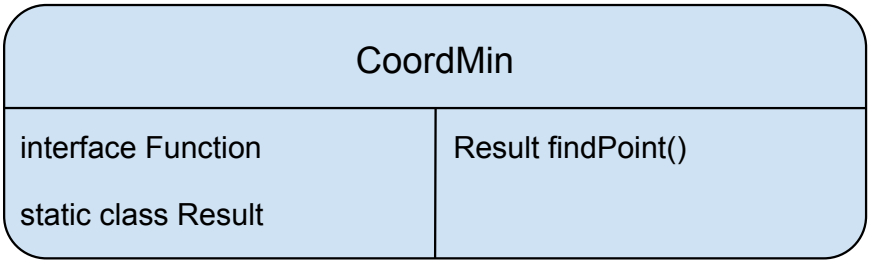

Рис. 13: Структура класса CoordMin.

SingularNode отражает простой маневр. Содержит в качестве полей желаемое состояние КА, модель, в которой производятся рассчеты, а также логические поля, показывающие, конец и невозможность выполнения маневра. Метод calculate() рассчитывает вектор тяги в случае требования на скорость, check() проверяет успешность выбранного вектора тяги, execute() исполняет маневр. Схема класса приведена на рисунке 14.

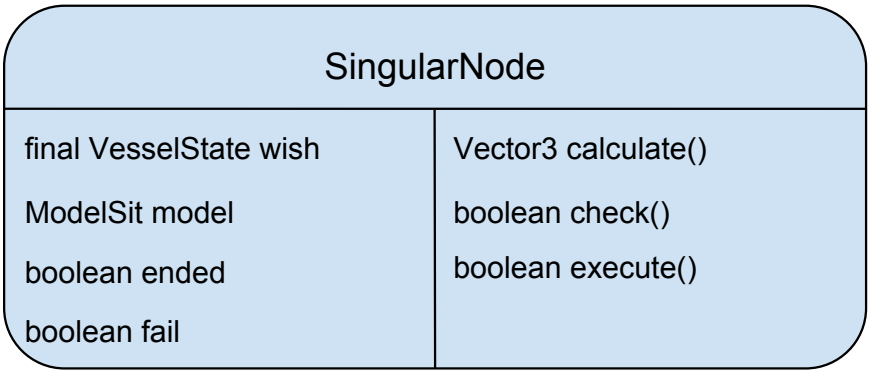

Рис. 14: Структура класса SingularNode.

LongSimpleManeuver выполняет такую же роль, что и SingularNode, но проверяет на успех по при помощи любой функции, используя интерфейс. На выходе выдает SingularNode, так что его схема здесь приведена не будет

Node series отражает сложный маневр, он представляет собой серию простых маневров, он хранит очередь из SingularNode как поле. Также

хранит в качестве полей модель, в которой происходят маневры, текущий простой маневр и логические поля, показывающие конец серии и невозможность выполнения. Серию невозможно выполнить, если хотя бы один из её простых маневров невозможно выполнить. Серия кончилась, если список простых маневров пуст. Метод execute() выполняет текущий простой маневр, если он кончился назначает новый текущий маневр из очереди, если очередь пуста, серия кончилась. Схема класса приведена на рисунке 15.

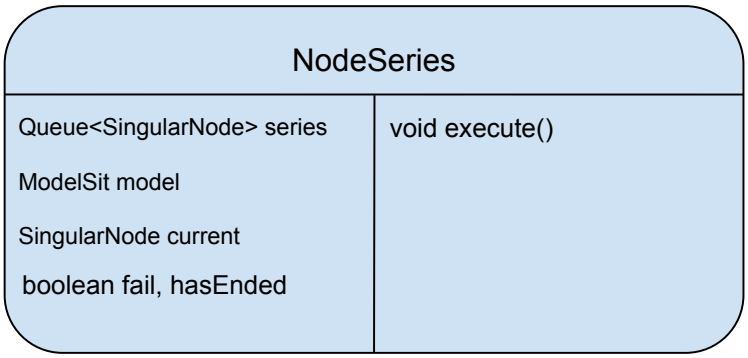

Рис. 15: Структура класса NodeSeries.

ManeuverPlanner представляет собой набор методов, каждый из которых отражает собой какой–либо конкретный маневр, конструируя его из простых требований и маневров. Также здесь происходит рассчет требований при необходимости. В виду множества методов, в схеме они приведены не будут. В качестве полей класс содержит реальную и две виртуальных модели для рассчетов, а также ссылки на КА в этих моделях. Схема класса приведена на рисунке 16.

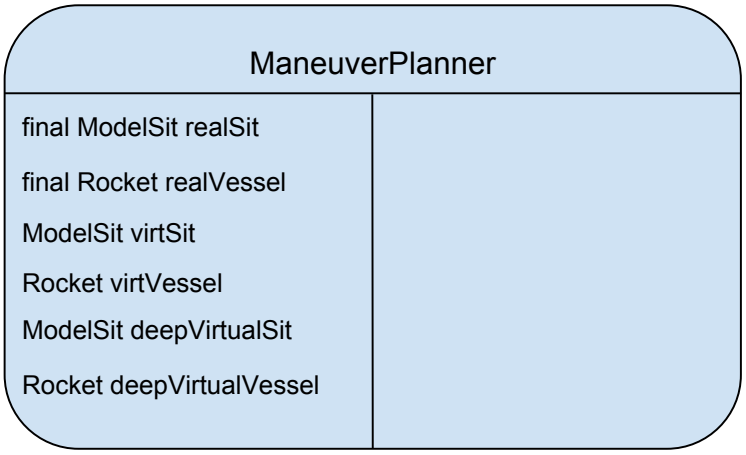

Рис. 16: Структура класса ManeuverPlanner.

Пакет graphics содержит единственный класс GMain, поэтому нет смысла приводить схему его структуры.

GMain — класс, содержащий метод main() программы, т.е. из него производится запуск. В качестве полей содержит параметры графического отображения — размеры окна, рисуемых объектов и т.д. (не отображены в схеме). Метод start() производит запуск графичекого отображения, и вызывает метод showSit(), который вызывает метод draw() в цикле, пока программа не будет остановлена. Также в начале метода вызывается iniSit(), задающий начальные данные, и описывается задача на управление КА. Метод draw() отображает на экране состояние системы с указанными в полях класса параметрами. Схема класса приведена на рисунке 17.

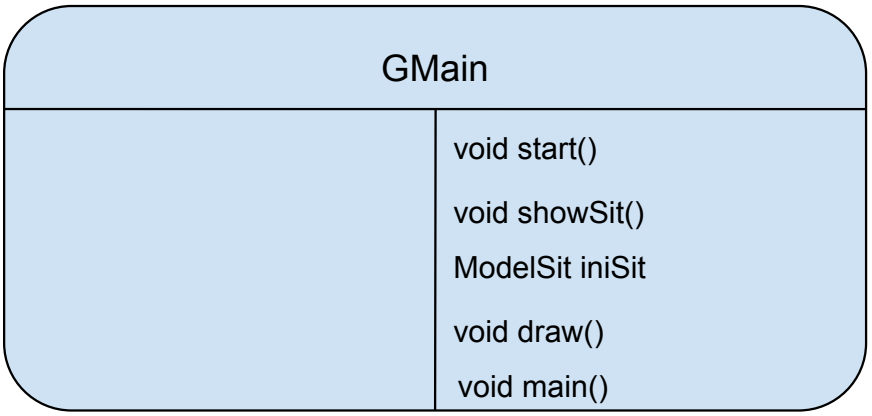

Рис. 17: Структура класса GMain.

На этом описание архитектуры программы можно считать законченным.

4.2. Примеры моделирования

Далее, на рисунках 18—23 представлены графические выводы программы при моделировании некоторых конкретных задач и начальных данных. Также, в приложении можно найти рисунки моделирования задачи N тел без КА. Приводится проекция на плосткость OXY (плоскость эклиптики), Желтый круг обозначает звезду, синие круги обозначают планеты, черная точка — КА. Размеры кругов не отражают радиус тел, масштаб выбран для наглядности. Красный пунктир — траектория КА и планет, в каждом случае траектория рисуется не для всех тел, а только для учавствующих в моделируемой задаче.

На рисунке 18 представлена картина поведения системы при отсутствии задачи на управление на КА, как следствие, при отсутствии управления на КА. Все тела в начале моделирования находятся на оси Ox, КА находится на высоте 152,2 млн километров, первая планета на высоте 249,232 млн километров, вторая на высоте 400 млн километров. Масса звезды равна массе Солнца, масса первой и второй планеты равны массе Марса. Все тела имеют начальную скорость, как необходимая скорость для круговой орбиты умноженная на 0,8.

На рисунке 19 можно наблюдать картину поведения системы при исполнении маневра по требованию на скорость в точке перицентра $\mathbf{v^u_{ka}}(t_{peri}) = \mathbf{v^0_{ka}}(t_{peri}) \cdot 0.6$. Начальные данные те же, что и в первом случае. Новая орбита имеет апоцентр высотой в 71,6234 млн километров, перицентр высотой в 23,2085 млн километров.

На рисунке 20 картина системы при исполнении сложного маневра по изменению апоцентра — сначала выход на круговую орбиту высотой 71,6234 млн, затем в нужный момент времени (297792–я минута от начала моделирования) изменение высоты апоцентра до 100 млн километров. Начальные данные те же.

На рисунках 21–22 представлено поведение системы при исполнении маневра по смене плоскости, в первом случае на плоскость с нормалью (1,0,1), во втором с нормалью (1,0,0). Начальные данные те же. В первом случае получается орбита с апоцентром и перицентром высотой соответственно 142,8962 и 80,9602 млн километров. Во втором случае получается орбита с перицентром 97,4447 млн километров и апоцентром 126,5010 млн километров.

На рисунке 23 отражено поведение системы при исполнении межпланетного перелёта. Планета старта отражает Землю, планета назначения — Марс, но орбиты у Марса круговая, а у Земли начальная скорость 0,8 от скорости, необходимой для круговой орбиты. Сначала выполняется выход на круговую орбиту высотой в 71,6234 млн километров, а затем совершается маневр по увеличению высоты апоцентра до требуемой (249,232 млн километров) в нужный момент(277372–я минута от начала моделирования). В точке соприкосновения ставится требование на скорость, компенсирующее разницу скоростей планеты и КА. В начале моделирования обе планеты

расположены на оси OX.

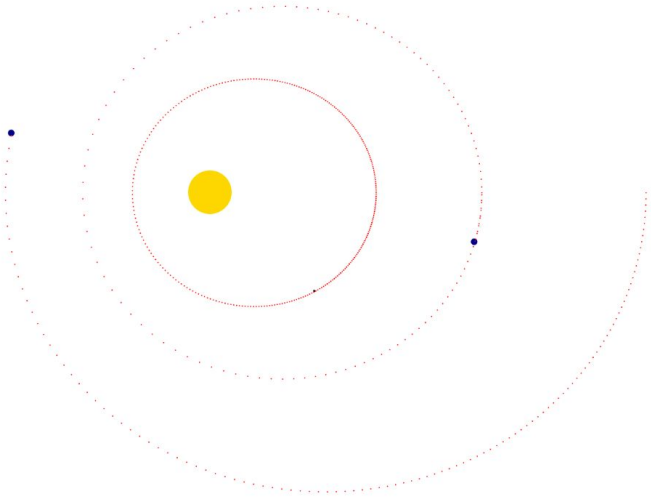

Рис. 18: Моделирование системы с нулевым управлением на КА.

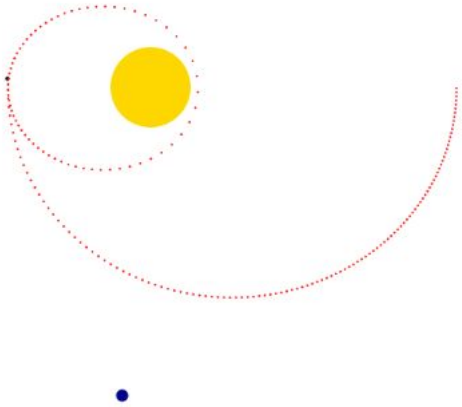

Рис. 19: Моделирование системы с маневром, удовлетворяющим требование на скорость.

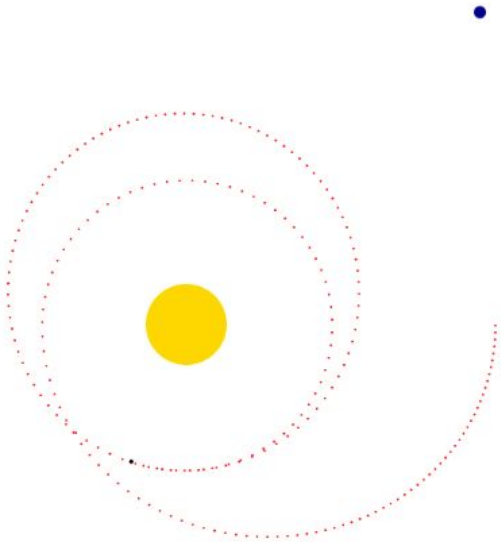

Рис. 20: Моделирование системы со сложным маневром по смене апоцентра.

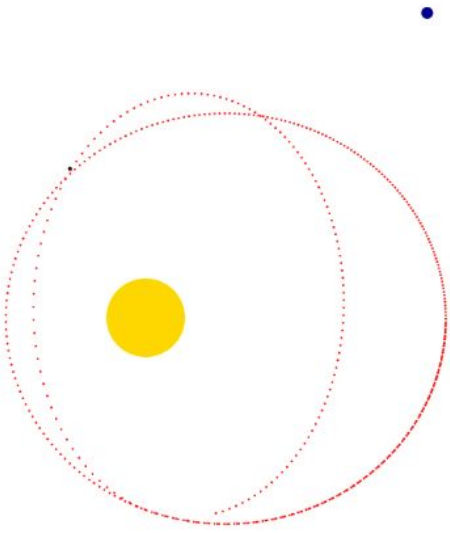

Рис. 21: Моделирование системы с маневром на изменение плоскости на плоскость с нормалью (1,0,1).

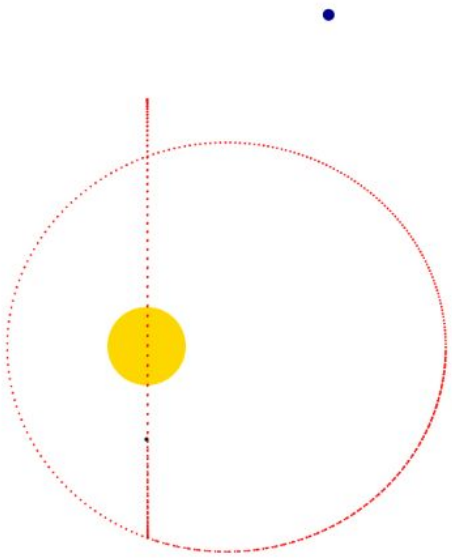

Рис. 22: Моделирование системы с маневром на изменение плоскости на плоскость с нормалью (1,0,0).

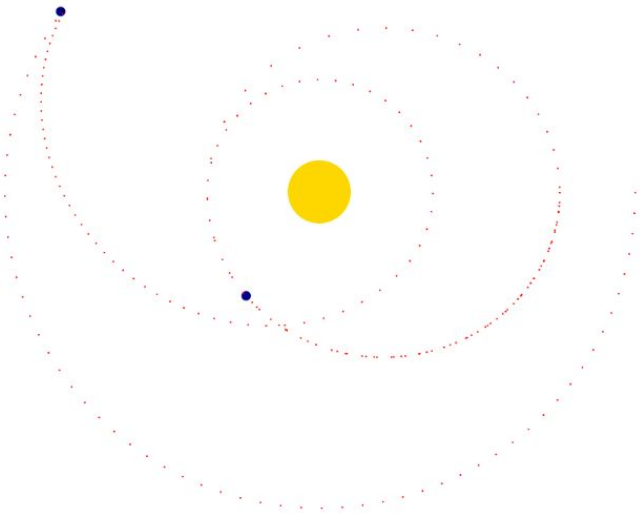

Рис. 23: Моделирование системы со сложным маневром по перелёту с одной планеты на другую.

Выводы

Целью работы являлось проведение моделирования динамики космичекого аппарата в поле звездной системы, тоесть проведение моделирования динамики N тел, одно из которых — звезда, имеющая массу много больше остальных тел, и ещё одно — КА, на который возможно подать управление.

В процессе решения поставленной задачи после анализа соответствующей литературы были сформированы система дифференциальных уравнений, описывающие динамику N тел без управления, и модифицированная система, позволяющая подавать управление на КА.

Для решения задач по удовлетворению требований по состоянию КА в нужный момент времени был предложен следующий подход: описать алгоритм выполнения маневра по удовлетворению требования на скорость КА в точке орбиты, который был назван простым маневром, и после этого представлять маневры по удовлетворению остальных требований как комбинацию нескольких простых маневров. Такие маневры были названы сложными. Самые короткие сложные маневры представляют собой комбинацию только простых маневров, более длинные — комбинацию простых и более коротких сложных маневров. Вычисление параметров маневра по требованиям предложено производить при помощи методов оптимизации, а именно минимизации функции отклонения состояния КА в нужный момент с текущим управлением от желаемого состояния КА в нужный момент времени.

Затем, на языке java с использованием платформы javaFX была разработанна компьютерная программа, реализующая моделирование и поиск управления с использованием описанных алгоритмов. Результаты моделирования свидетельствуют об эффективности реализованного подхода.

В то же время работа требует серьёзного развития по следующим направлениям:

1) В реальных условиях важно не только исполнение заданного требования, но также затраченное время и средства, в связи с чем необходимо также проводить минимизацию по затраченному времени и количеству топлива. Для этого необходимо оптимизировать предложенные алгоритмы, так как в текущем виде необходимая минимизация будет

занимать слишком большое количество времени, что делает невозможным тестирование, и , как следствие, разработку программы.

2) Для приближения модели к реальным условиям, в особенности при расчетах межпланетных маневров, необходимо учитывать атмосферу и вращение планет вокруг своей оси, также возможно развитие модели по другим направлениям

3) Полная информация о системе в реальном случае может оказаться недоступной, поэтому необходимо реализовать механизм сбора доступной информации, и на основе её, расчёта параметров системы

Заключение

В ходе выполнения данной работы были получены следующие основные результаты результаты:

1) На основе анализа соответствующей литературы сформирована система дифференциальных уравнений, описывающая динамику тел в задаче N тел с обеспечением механизма подачи управления на КА

2) Выбран и описан алгоритм решения полученной системы уравнений

3) Предложены и реализованны алгоритмы поиска управления, позволяющие решать задачу удовлетворения требований на скорость КА, на параметры орбиты, на координату и скорость КА в нужный момент времени, а также осуществлять межпланетные перелёты

4) Разработана компьютерная программа, реализующая моделирование и поиск управления по предложенным алгоритмам

Список литературы

1. *Аббасов М. Э.*, Методы оптимизации: Учеб. пособие— СПб.: Издательство “ВВМ”, 2014.

2. *Авдюшев В. А.*, Интегратор Гаусса-Эверхарта. Томск 2010

3. *Авдюшев В. А.*, Численное моделирование орбит небесных тел. Томск, Издательский Дом Томского государственного университета 2015

4. *Ермолин В. С., Королев В. С., Потоцкая И. Ю.*,Теоретическая механика. Часть 2. Динамика. Учебное пособие.—СПб: СПбГУ, ВВМ, 2013.

5. *Иванов Н. М.,Лысенко Л. Н*, Баллистика и навигация космических аппаратов. Москва 2004

6. *Иванов А. П.*, Решение систем линейных алгебраических уравнений. Санкт–Петербург 2016

7. *Мартынова А. И., Орлов В. В., Рубинов А. В.*, Динамика тройных систем: Учеб. пособие. — СПб.: Изд-во С.-Петерб. ун-та, 2010.

8. *Олемской И. В.*, Численные методы. Часть 2. Учебное пособие. Санкт-Петербург 2013

9. *Раушенбах Б. В.*, Управление движением космических аппаратов. Издательство «Знание» Москва 1986

10. *Рой А.*, Движение по орбитам. М.: Мир,1981

11. *Bruce Eckel*, Thinking in Java 4th edition. Prentice Hall 2006

12. *Steve Mcconnell*, Code Complete, 2nd Edition. Wiley India Pvt. Limited 2004

13. *Java documentation*, Java SE javaFX Canvas API, `https://docs.oracle.com/javafx/2/canvas/jfxpub-canvas.htm`

Приолжение

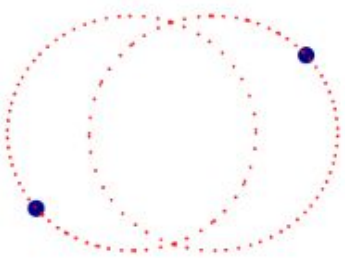

Рис. 24: Моделирование двух одинаковых тел, расположенных симметрично, с одиноковымыми начальными скоростями.

Рис. 25: Моделирование двух тел с разной массой, расположенных симметрично, с одиноковымыми начальными скоростями.

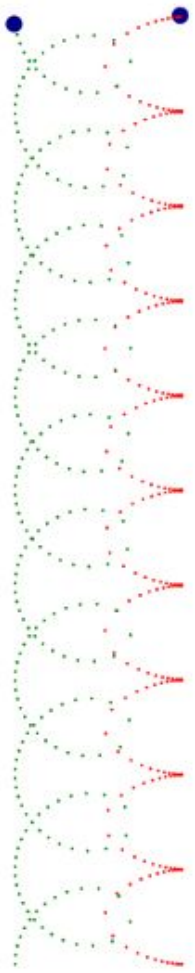

Рис. 26: Моделирование двух тел с разной массой, расположенных симметрично, с нулевой скоростью у более массивного тела.

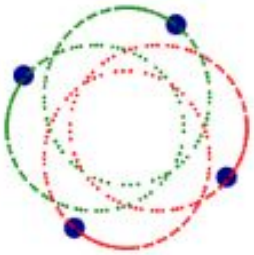

Рис. 27: Моделирование четырёх одинаковых тел, расположенных симметрично, с одиноковымыми начальными скоростями.

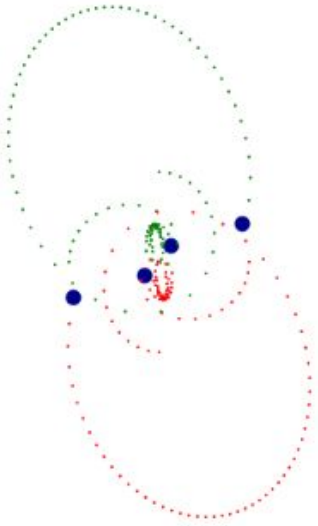

Рис. 28: Моделирование четырёх тел с попарно одинаковой массой, расположенных симметрично, с одиноковымыми начальными скоростями.

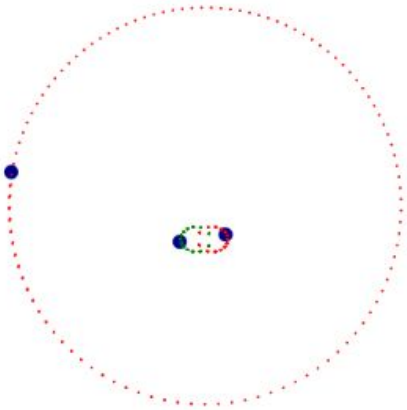

Рис. 29: Моделирование планеты на орбите двойной звезды.