Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский физико-технический институт
(национальный исследовательский университет)»
Физтех-школа Аэрокосмических Технологий
Кафедра математического моделирования и прикладной математики

Направление подготовки / специальность: 03.03.01 Прикладные математика и физика
Направленность (профиль) подготовки: Геокосмические науки и технологии

ПОИСК БЕЗОПАСНЫХ ОРБИТ ДЛЯ ИЗБЕГАНИЯ СТОЛКНОВЕНИЯ С ОБЪЕКТАМИ В ОКОЛОЗЕМНОМ ПРОСТРАНСТВЕ

(бакалаврская работа)

Студент:
Забара Илья Денисович

_____
(подпись студента)

Научный руководитель:
Широбоков Максим Геннадьевич,
канд. физ.-мат. наук

_____
(подпись научного руководителя)

Консультант *(при наличии)*:
_____
_____
(подпись консультанта)

Москва 2024

Аннотация

С каждым днём обеспечение безопасности космических аппаратов становится всё более актуальной проблемой.

Настоящая работа состоит состоит из двух основных частей. В первой части ставится и решается задача оптимизации поиска наиболее близкой к исходной (геометрически) и безопасной орбиты в целях избегания столкновения при заданных исходной орбите, множества опасных орбит и минимально необходимого расстояния до всех опасных орбит. Описывается применяемый метод внешних штрафных функций для её решения. Вторая часть работы состоит в определении связи между навигационной неопределенностью опасной орбиты (среднеквадратичными отклонениями по положению и скорости) и необходимым минимальным расстоянием до неё.

Для численного решения всех возникающих в процессе задач реализован программный инструмент `collisionAvoidance` (на языке Python), использующий сторонние общедоступные библиотеки.

Оглавление

Введение

В области современной астродинамики актуальной является задача обеспечения безопасности космических аппаратов. Эта проблема становится всё более важной на фоне растущего количества космического мусора — отработанных ступеней ракет, уже неработающих спутников, обломков. Космический мусор представляет собой серьёзную угрозу для спутников и прочих космических аппаратов с полезной нагрузкой, так как даже небольшие объекты, двигаясь с высокой относительной скоростью, способны нанести значительный ущерб или полностью вывести из строя космическую технику.

В подавляющем большинстве работ по тематике обеспечения безопасности полётов рассматривается именно специфика маневрирования для избегания столкновения. Оптимизируемые функционалы, как правило, связаны с импульсными затратами топлива для перехода на ту или иную орбиту. Кратко изложим результаты нескольких таких работ.

Работа [1] посвящена оптимизации импульсных маневров для избегания столкновений на низких околоземных орбитах. Авторы представили аналитическую формулировку для вычисления расстояния промаха и вероятности столкновения двух объектов при выполнении импульсивного манёвра уклонения. Они использовали линейную зависимость между приложенным импульсом и относительным движением объектов, что позволило сформулировать задачу оптимизации манёвра как задачу на собственные значения, связанную с простым нелинейным алгебраическим уравнением. Цель оптимизации заключалась в минимизации затрат манёвра в терминах величины характеристической скорости ΔV для максимизации расстояния промаха или минимизации вероятности столкновения. На завершающих этапах работы предложен численный метод решения задачи оптимизации в самой общей форме, используя полученные в ходе работы аналитические выражения. Похожий подход продемонстрирован в более новой работе [2]

от других авторов.

В работе [3] проблема избегания столкновения с объектами в околоземном пространстве формулируется как задача многокритериальной оптимизации, где основными целями минимизации являются характеристическая скорость, вероятность столкновения и расстояние между спутником и опасным объектом после маневра. В работе предложена методика, которая позволяет учитывать множество сближений в краткосрочной перспективе, что является критически важным, например, для геостационарной орбиты (GEO). Авторы используют алгоритм многокритериальной оптимизации с использованием метода роя частиц, который позволяет вычислить набор оптимальных по Парето решений. Инструментом для оценки целевой функции и вычисления вероятности столкновения для отобранных оптимальных по Парето сценариев является глобальный оптимизатор COSY-GO. Приводятся результаты для некоторых типичных орбит. Результаты работы показывают, что учёт всех возможных сближений после исполнения маневра уклонения от опасного объекта может предотвратить возникновение новых опасных сближений в краткосрочной перспективе.

В статье [4] также рассматривается оптимизация манёвров по избеганию столкновений. Манёвры моделируются как многократные импульсы. Оптимизация формулируется в виде последовательности задач выпуклой оптимизации, решаемых методом внутренней точки. Основной целью в работе является также минимизация топлива, используемого для результирующего манёвра. В результате разработан метод, рассчитывающий оптимальные манёвры без предположений о структуре и направлении импульса.

В работе [5] описывается специализированное программное обеспечение, предназначенное для вычисления манёвров предотвращения столкновений с минимальным расходом топлива в краткосрочных сценариях встречи. В статье также рассматриваются несколько методов расчёта вероятности столкновения.

Немного иной подход продемонстрирован в статье [6]. Основное внимание в работе уделяется использованию машинного обучения для предсказания рисков столкновения на основе большого набора данных, собранного Европейским космическим агентством в период с 2015 по 2019 год.

В работе [7] рассматривается создание автономной траектории для вывода спутников с орбиты с учётом избегания столкновений, то есть рассматриваемая в этой статье задача в каком-то смысле является обратной, что означает то, что ищется траектория именно для выводимого с орбиты аппарата. Делается это в целях сохранить остальные аппараты при пролёте по этой траектории. При этом снова, как и в остальных работах, оптимизировались топливные затраты, но уже с применением принципа максимума Понтрягина и синусоидального метода (он основан на простой гармонической функции и представляет траекторию в виде кусочно-синусоидальной волны). Результаты указывают на необходимость совершенствования методов избегания столкновений при автономной траектории вывода спутников с орбиты, что важно для обеспечения устойчивого космического окружения.

К вопросу об актуальности, в электронном ресурсе [8] Европейским Космическим Агенством был рассмотрен вопрос возрастающей вероятности столкновений с космическим мусором. Как и было указано в мотивации, действительно, вероятность столкновений в космосе и образования нового космического мусора стремительно возрастает. Был опубликован доклад об экономических издержках, связанных с космическим мусором. Космический мусор обходится дорого и в будущем станет ещё дороже. Упомянуты ведущиеся разработки ЕКА для автоматического предотвращения столкновений.

В нашем исследовании такая задача будет рассмотрена немного с другой стороны. Предлагается геометрический подход к определению безопасных орбит. А именно, предлагается абстрагироваться от того, в какие моменты и с какой величиной производятся импульсы скорости, и поставим

оптимизационную задачу поиска орбиты, которая, во первых, минимально отличалась бы от исходной орбиты, и, во-вторых, обеспечивала бы заданное минимальное расстояние до опасных орбит. В этом есть весьма определённая мотивация: минимальное изменение орбитальных элементов обусловлено необходимостью сохранения возможности удовлетворения целей миссии, функциональности полезной нагрузки космического аппарата, включая работу научного оборудования, систем связи и навигации. Большое отклонение от исходной орбиты в целях избегания столкновений может привести к некорректной работе определённого оборудования, которое, зачастую, расчитано на конкретную орбиту. Таким образом, наш подход отличается от остальных в первую очередь тем, что *ищется не импульс, а орбиту.*

Немаловажной частью данной работы также будет являться решение вопроса о том, на какое расстояние стоит «увести» наш аппарат от остальных, чтобы новую орбиту можно было в действительности считать безопасной. Предлагается рассматривать данный вопрос опираясь на навигационную неопределенность положений и скоростей объектов на опасных орбитах. Основным подходом к решению этой задачи будет является использование метода автоматического дифференцирования, что в итоге позволит эффективно и с высокой точностью аппроксимировать решение задачи Коши относительно начальных условий формулой Тейлора произвольного порядка.

Таким образом, целью настоящей работы является разработка и реализация собственной методики определения наиболее близких к исходной безопасных орбит в околоземном пространстве для предотвращения столкновений с космическими объектами с учётом их навигационной неопределённости.

1. Оптимизационная задача поиска безопасной и наиболее близкой к исходной орбиты космического аппарата

Пусть даны начальная орбита космического аппарата, а также известный набор орбит космического мусора. Итак, как уже было отмечено во введении, стоит задача разработки методики поиска орбиты, которая была бы максимально приближена к исходной и в то же время безопасной, то есть находящейся вне зоны возможного столкновения с космическим мусором. Для этого необходимо ввести ключевые понятия и математические операции, которые будут рассмотрены в первых трёх параграфах этой главы. Они будут посвящены задаче поиска минимального расстояния между кеплеровскими эллипсами, а также выбору целевой функции задачи оптимизации.

1.1. Расстояние между кофокальными эллипсами в пространстве

Для начала будем рассматривать простейшую модель движения аппарата на околоземной орбите — задачу двух тел. В этой модели такие орбиты, как известно, являются замкнутыми линиями — эллипсами, в фокусе которых располагается земной центр масс. В дальнейшем, на основе результатов, полученных на этой модели, рассмотрим более сложную модель движения, учитывающую зональную гармонику J_2.

Перед постановкой основной оптимизационной задачи, как уже было отмечено, следует ввести понятия, которые будут задействованы в ходе её постановки. Одним из таких понятий является *расстояние между двумя кофокальными эллипсами в пространстве*. Введём это понятие.

Введём классическую правую прямоугольную систему координат с началом в притягивающем центре Земли. В таком случае орбиты будем задавать векторами $\boldsymbol{x} = (a, e, i, \omega, \Omega)^T$, представляющими из себя набор пяти орбитальных элементов, определяющих форму и положение кеплеровского эллипса в пространстве. Здесь a, e, i, ω и Ω – большая полуось, эксцентриситет, наклонение, аргумент перицентра и долгота восходящего узла соответственно. Расстояние между двумя орбитами $\rho(\boldsymbol{x}, \boldsymbol{x'})$ будем понимать в следующем смысле: минимальное значение расстояний между двумя точками, лежащими на двух заданных эллипсах. Всё, что относится ко второй орбите, далее будет отмечено штрихом. Так как нас интересует вся орбита целиком, то значение эксцентрической аномалии u для каждой орбиты пробегает все возможные значения в диапазоне $[0, 2\pi)$, а сама орбита может быть однозначно задана набором пяти орбитальных элементов: $\boldsymbol{x} = (a, e, i, \omega, \Omega)^T$. Радиус-вектор на произвольной орбите, определённой набором орбитальных элементов $(a, e, i, \omega, \Omega)$, может быть задан так:

$$\frac{\boldsymbol{r}}{a} = \boldsymbol{P}(\cos u - e) + \boldsymbol{S} \sin u, \tag{1}$$

где $\boldsymbol{S} = \sqrt{1 - e^2}\boldsymbol{Q}$, а компоненты векторов $\boldsymbol{P}$ и $\boldsymbol{Q}$

$$\begin{aligned}
P_x &= \cos\omega\cos\Omega - \cos i \sin\omega \sin\Omega, \\
P_y &= \cos\omega\sin\Omega - \cos i \sin\omega \cos\Omega, \\
P_z &= \sin i \sin\omega, \\
Q_x &= -\sin\omega\cos\Omega - \cos i \cos\omega \sin\Omega, \\
Q_y &= -\sin\omega\sin\Omega + \cos i \cos\omega \cos\Omega, \\
Q_z &= \sin i \cos\omega.
\end{aligned} \tag{2}$$

Таким образом, для двух фиксированных орбит с заданными элементами расстояние между ними будет функцией соответствующих аргументов

u. Для нормированного квадрата расстояния $\hat{\rho}(u,u') = |\boldsymbol{r} - \boldsymbol{r}'|^2/2aa'$, согласно [9], справедлива следующая формула:

$$\begin{aligned}\hat{\rho}(u,u')\big|_{\boldsymbol{x},\boldsymbol{x}'} = \hat{\rho}_0 &+ (\boldsymbol{P}^T\boldsymbol{P}' - \alpha e)\cos u + \boldsymbol{S}^T\boldsymbol{P}'e'\sin u + \\ &+ (\boldsymbol{P}^T\boldsymbol{P}'e' - \alpha' e')\cos u' + \boldsymbol{P}^T\boldsymbol{S}'e\sin u' - \\ &- \boldsymbol{P}^T\boldsymbol{P}'\cos u\cos u' - \boldsymbol{P}^T\boldsymbol{S}'\cos u\sin u' - \\ &- \boldsymbol{S}^T\boldsymbol{P}'\sin u\cos u' - \boldsymbol{S}^T\boldsymbol{S}'\sin u\sin u' + \\ &+ \frac{\alpha}{4}e^2\cos 2u + \frac{\alpha'}{4}e'^2\cos 2u',\end{aligned} \tag{3}$$

где $\hat{\rho}_0 = \dfrac{\alpha+\alpha'}{2} + \dfrac{\alpha e^2 + \alpha' e'^2}{4} - \boldsymbol{P}^T\boldsymbol{P}'ee'$, $\alpha = \dfrac{a}{a'}$, $\alpha' = \dfrac{a'}{a}$.

Расстояние $\rho(\boldsymbol{x},\boldsymbol{x}')$ между двумя орбитами, понимаемое как минимальное значение расстояний между двумя точками, лежащими на двух заданных эллипсах, является глобальным минимумом функции (3) для значений аргументов, принадлежащих тору $[0,2\pi]\times[0,2\pi]$. То есть, чтобы найти значение нужной функции $\rho(\boldsymbol{x},\boldsymbol{x}')$, нужно найти минимум функции $\hat{\rho}(u,u')\big|_{\boldsymbol{x},\boldsymbol{x}'}$ на заданном множестве. То есть, для двух фиксированных орбит, математически расстояние между ними выражается по формуле:

$$\rho(\boldsymbol{x},\boldsymbol{x}') = \min_{(u,u')\in[0,2\pi)\times[0,2\pi)} \hat{\rho}(u,u')\,\big|_{\boldsymbol{x},\boldsymbol{x}'} \tag{4}$$

В той же работе [9] показывается, что нахождение экстремальных точек функции (3) можно свести к определению всех действительных корней некоторого многочлена. Коэффициенты многочлена являются рациональными функциями параметров орбиты. Полином как функция одной переменной u выглядит следующим образом:

$$\begin{aligned}g(u) = K^2C^4 - N^2A^2 - M^2B^4 &+ 2NKCA(A^2 - C^2) + \\ &+ 2NMBA(A^2 + B^2) + 2KMCB(B^2 - C^2) + \\ &+ (M^2 + N^2 - K^2)(C^2A^2 - A^2B^2 + C^2B^2),\end{aligned} \tag{5}$$

где

$$
\begin{aligned}
A &= \boldsymbol{P}^T\boldsymbol{S}'\sin u - \boldsymbol{S}^T\boldsymbol{S}'\cos u, \qquad B = \boldsymbol{P}^T\boldsymbol{P}'\sin u - \boldsymbol{S}^T\boldsymbol{P}'\cos u,\\
C &= (\boldsymbol{P}^T\boldsymbol{P}e' - \alpha e)\sin u + \alpha e^2 \sin u\cos u - \boldsymbol{S}^T\boldsymbol{P}'e'\cos u,\\
M &= \boldsymbol{P}^T\boldsymbol{P}'\cos u + \boldsymbol{S}^T\boldsymbol{P}'\sin u + \alpha' e' - \boldsymbol{P}^T\boldsymbol{P}'e,\\
N &= \boldsymbol{P}^T\boldsymbol{S}'e - \boldsymbol{S}^T\boldsymbol{S}'\sin u - \boldsymbol{P}^T\boldsymbol{S}'\cos u, \qquad K = \alpha' e'^2.
\end{aligned}
\tag{6}
$$

При этом каждому найденному корню u полинома (5) ставится в соответсвие величина u', кототая находится из следующих уравнений:

$$
\begin{aligned}
\cos u' &= \frac{BC + kA\sqrt{A^2 + B^2 - C^2}}{A^2 + B^2},\\
\sin u' &= \frac{AC - kB\sqrt{A^2 + B^2 - C^2}}{A^2 + B^2},
\end{aligned}
\tag{7}
$$

где величина k равна плюс или минус единице. Выбор знака k происходит так, чтобы было выполнено соотношение $M\sin u' + N\cos u' = K\cos u'\sin u'$.

Учитывая это, рассмотрим несколько подходов к поиску расстояния между двумя кеплеровыми кофокальными эллипсами на практике. Первый способ будет основан на применении описанной выше теории, связанной с поиском нулей полинома (5). Для примера возьмём такие две орбиты:

	a, тыс. км	e	i	ω	Ω
$\boldsymbol{x}$	7.13	0.21	64.4°	169°	50°
$\boldsymbol{x}'$	9.83	0.34	88.4°	26°	28°

Таблица 1 – Пример двух орбит для поиска расстояния между ними.

График полинома $g(u)$ для таких двух орбит на отрезке $[0; 2\pi]$ выглядит следующим образом:

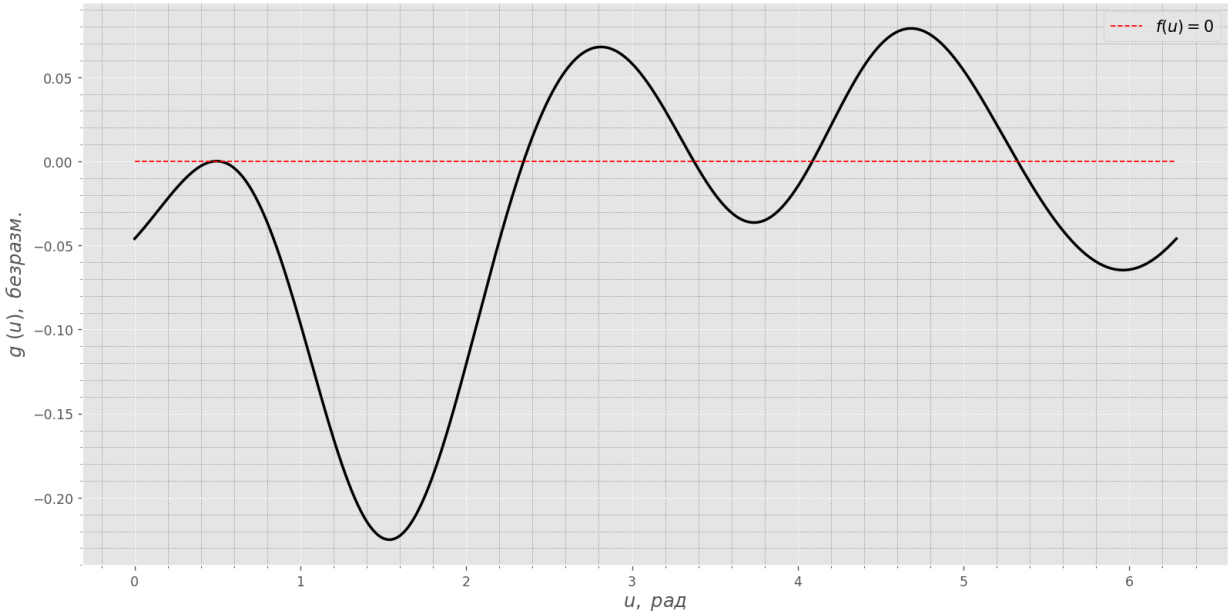

Рисунок 1 – График полинома $g(u)$.

Рассмотрим следующую процедуру. Стротся достаточно мелкое разбиение отрезка $[0, 2\pi]$ и на каждом из подотрезков ищем корни методом дихотомии. При этом сохраняем уникальные найденные корни и для каждого из них (и соответствующей ему пары u') считаем значение функции (3), затем выбираем минимум получившегося массива. Этот метод назовём *методом последовательной дихотомии.*

Как альтернативу рассмотрим ещё несколько подходов, один из которых основан на поиске минимума (3) с помощью численных методов оптимизации. Нетрудно показать, что (3) не является выпуклой и многоэкстримальна на множестве $[0, 2\pi] \times [0, 2\pi]$. Так как интересен именно глобальный минимум функции (3), то будем рассмтривать методы глобальной оптимизации. Будем использовать два метода: метод имитации отжига и метод дифференциальной эволюции. Сама функция на рассматриваемом множестве выглядит следующимм образом:

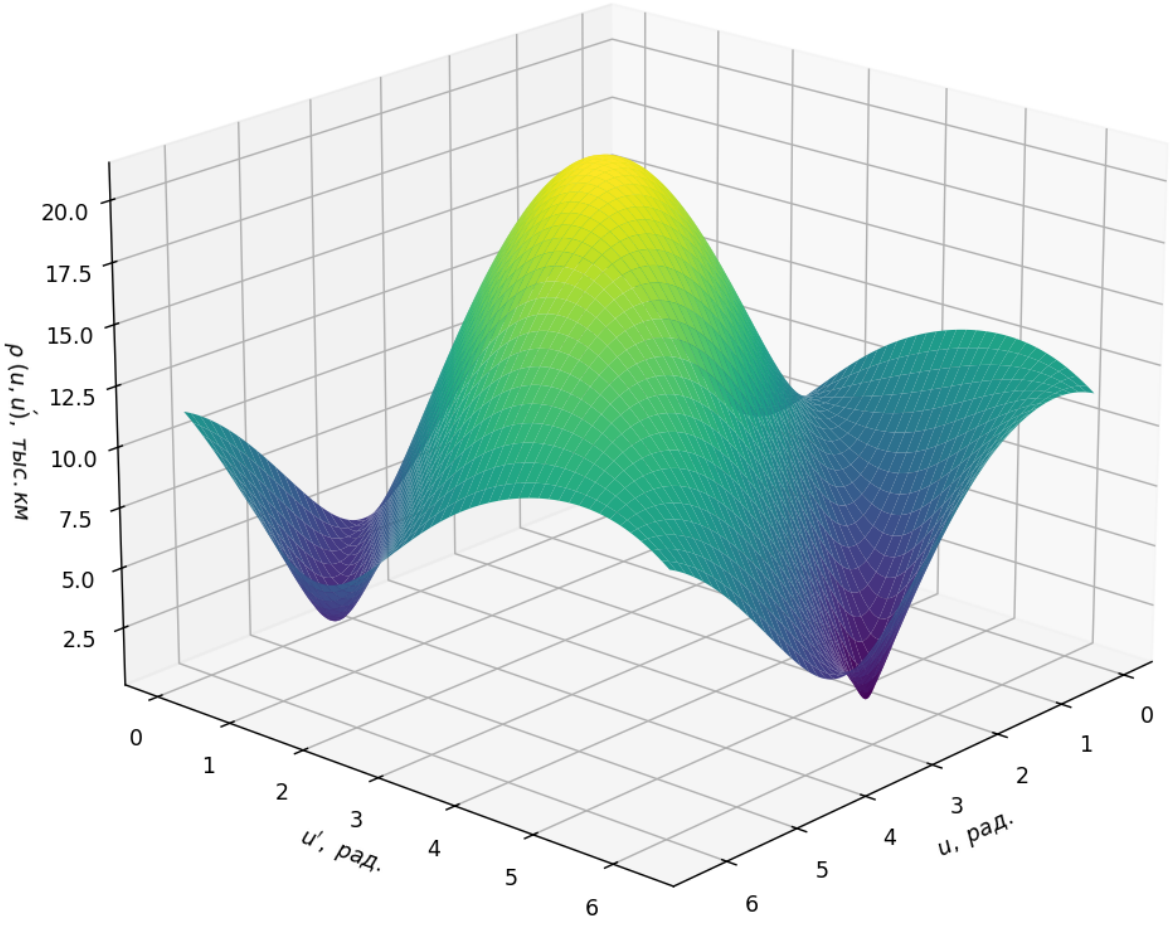

Рисунок 2 – График функции $\hat{\rho}(u, u')$.

Кроме того, рассмотрим метод локальной оптимизации для оценки его вычислительной эффективности. Такой метод может сойтись только лишь к локальному минимуму, и нет гарантии, что значение функции (3) в этой точке будет минимально возможным. Однако, если в будущем удастся разработать методику выбора подходящего начального приближения для данной задачи, такие методы могут оказаться эффективными и применимыми в данной задаче. Будем использовать метод Нелдера–Мида [10].

Ещё одна предлагаемая методика основана на дискретизации орбит и представлением их в виде набора точек в количестве N штук. Процедура следует определению функции $\rho(\boldsymbol{x}, \boldsymbol{x}')$: вычисляем расстояние между каждой точкой одной орбиты и каждой точкой другой орбиты, а затем ищем минимальное значение из полученного множества. Такой способ будем

называть *нативной процедурой*. Математически это можно записать так:

$$\rho_D(\boldsymbol{x}, \boldsymbol{x}') = \min_{k,l \in 1,\ldots,N} \|\boldsymbol{R}(\boldsymbol{x}, u_k) - \boldsymbol{R}(\boldsymbol{x}', u_l')\|_2, \tag{8}$$

где $\boldsymbol{R}(\boldsymbol{x}, u)$ – функция, ставящая в соответствие орбитальным элементам и эксцентрической аномалии радиус-вектор соответствующей точки.

Проведённые в данной работе эксперименты показали, что данный простой метод демонстрирует высокую эффективность и в дальнейшем будет является основным при решении задачи оптимизации.

Применим все описанные выше методы к настоящей задаче и получим результаты, представленные в Таблице 2.

Метод	**Расстояние между орбитами, км**	**Среднее время расчёта, мс**
Нативная процедура дискретизации, $N = 200$	526.5	16
Нативная процедура дискретизации, $N = 500$	525.2	46
Исследование нулей $g(u)$ методом последовательной дихотомии	523.7	29
Метод Нелдера–Мида	2984.5	8
Метод имитации отжига	523.7	373
Метод дифференциальной эволюции	523.7	82

Таблица 2 – Сравнение различных методов дискретизации и оптимизации.

Истинное значение расстояния между рассматриваемыми орбитами равно 523.7 км. Видим, что метод последовательной дихотомии и методы глобальной оптимизации сошлись к истинному значению. Нативная процедура дискретизации дала приближённое значение, но с достаточно хорошей точностью.

Методы глобальной оптимизации дают верный ответ, но работают долго. Метод Нелдера–Мида работает быстрее всех методов, но сходится лишь к локальному минимуму. Дальнейшее усовершенствование алгоритма

поиска расстояния между двумя орбитами может быть связано с правилом выбора правильного начального приближения для метода локальной оптимизации.

Посмотрев на значения среднего времени расчёта для каждого метода в Таблице 2, делаем вывод: если требуется большая точность, то применяем метод последовательной дихотомии, но при этом в ~ 2 раза увеличивается среднее время расчёта в сравнении с методом дискретизации орбит при $N = 200$. Повышать количество точек дискретизации не имеет смысла. Теперь перейдём к выбору целевой функции оптимизационной задачи.

1.2. Выбор целевой функции

В этом разделе осуществим выбор целевой функции, которая в дальнейшем станет целевой функцией основной задачи оптимизации. Как уже упоминалось ранее, целью является поиск наболее близкой орбиты с геометрической точки зрения. Функция (4), рассмотренная в предыдущем параграфе, не является подходящей для этого, поскольку, к примеру, её нулевое значение при пересечении двух орбит не гарантирует их идентичность. Это обусловлено тем, что пересекающиеся орбиты в общем случае могут существенно различаться по размерам и пространственной ориентации. В таком случае часто вводят взвешенную норму разности орбитальных элементов двух орбит ($\boldsymbol{x}$ и $\boldsymbol{x}_0$) как функцию переменной $\boldsymbol{x}$ при известной начальной орбите $\boldsymbol{x}_0$:

$$f(\boldsymbol{x}, \boldsymbol{x}_0) = \sum_{i=1}^{5} w_i (x_i - x_{0i})^2, \tag{9}$$

где w_i – вес i-го орбитального элемента, а x_i – сам орбитальный элемент. Соответствие между индексом i и орбитальным элементом будем использовать такое:

$$\boldsymbol{x} = (a, e, i, \omega, \Omega)^T = (x_1, x_2, x_3, x_4, x_5)^T.$$

В астродинамике такую функцию, сравнивающую близость двух орбит, обычно называют *D-критерием* [11]. Это естественный способ сравнивать геометрически две орбиты между собой. Однако, у (9) есть недостаток, связанный с тем, что для угловых орбитальных элементов разницы (5°− 0°) и (355°− 0°) имеют на порядки отличающиеся числовые значения, но с геометрической точки зрения эти случаи не отличаются. Поэтому рассмотрим следующую модификацию функции (9):

$$f(\boldsymbol{x}, \boldsymbol{x}_0) = \sum_{i=1}^{2} w_i (x_i - x_{0i})^2 + \sum_{i=3}^{5} w_i \left[(\sin x_i - \sin x_{0i})^2 + (\cos x_i - \cos x_{0i})^2 \right]. \tag{10}$$

Использование (10) вместо (9) в качестве целевой функции приводит к утрате свойства выпуклости, что имеет существенное значение в контексте оптимизации.

В литературе также упомянаются другие естественные метрики в пространстве эллиптических орбит. В работе [12] приведена следующая 2-норма для геометрического сравнения двух орбит $\boldsymbol{x}$ и $\boldsymbol{x}'$:

$$d^2(\boldsymbol{x}, \boldsymbol{x}') = \left(W_0 - \sqrt{(W_5 + W_8)^2 + (W_6 - W_7)^2} \right), \tag{11}$$

где

$$4W_0 = 2(\alpha + \alpha') + \alpha e^2 + \alpha' e'^2 - 4\boldsymbol{P}^T\boldsymbol{P}', \quad 2W_5 = -\boldsymbol{P}^T\boldsymbol{P},$$
$$2W_6 = -\boldsymbol{P}^T\boldsymbol{S}', \quad 2W_7 = -\boldsymbol{S}^T\boldsymbol{P}', \quad 2W_8 = -\boldsymbol{S}^T\boldsymbol{S}'.$$

Авторами делается упор на то, что D-критерий является менее строгой с математической точки зрения метрикой, чем (11), которая, по авторскому определению в [12], имеет смысл интегрального среднего расстояния между

двумя орбитами. Авторами показывается, что (11) удовлетворяет аксиомам метрического пространства, в то время как для D-критерия подобных выводов в литературе обнаружить не удалось. Однако, у D-критерия есть возможность контролировать «важность» орбитальных элементов, определяя веса w_i, в отличие от (11). Также стоит обратить внимание, что (11) и функция (4), введённая в п.1.1, имеют различный смысл. Выражение (11) может быть целевой функцией и отражать «близость» двух орбит друг к другу, она, как уже было упомянуто, является интегральным средним функции (3). В то же время, функция (4) в свою очередь, является глобальным минимумом функции (3) на множестве $[0, 2\pi) \times [0, 2\pi)$. И самое главное то, что (11) является явной формулой для расчёта, а (4) – нет, для вычисления значения этой функции приходится использовать процедуры, описанные в п.1.1.

В проведённых численных экспериментах при решении поставленной далее оптимизационной задачи в целях сравнения испытывались все 3 введённые выше нормы. Во всех трёх случаях при равных весах орбитальных элементов в D-критериях метод сходился к одному и тому же решению при идентичных начальных данных и начальном приближении. В дальнейшем основным функционалом будет (10).

1.3. Постановка оптмизационной задачи

Итак, в предыдущих параграфах были введены все понятия, необходимые для постановки задачи поиска наиболее близкой и безопасной орбиты космического аппарата. Теперь можем привести саму постановку:

$$\begin{cases} f(\boldsymbol{x}, \boldsymbol{x}_0) \longrightarrow \min\limits_{\boldsymbol{x} \in \Pi} \\ r_{\min} - \rho(\boldsymbol{x}, \boldsymbol{x}_j) \leqslant 0, \quad j = [1, \ldots, m] \end{cases}, \tag{12}$$

где целевой функцией $f(\boldsymbol{x}, \boldsymbol{x}_0)$ является функция (10), $\boldsymbol{x}_0$ – начальная орбита космического аппарата, $r_{\min}$ – заданный параметр безопасного рас-

стояния, $\{\boldsymbol{x}_j\}_{j=1}^m$ – каталог орбит (наборов орбитальных элементов) опасных объектов, введённая уже ранее функция $\rho(\boldsymbol{x}, \boldsymbol{x}_j)$ является расстоянием в координатном пространстве между ближайшими точками двух орбит, а принадлежность оптимизационной переменной множеству

$$\Pi = \{\boldsymbol{x} \in \mathbb{R}^5 | x_1 > 0, x_2 \in (0, 1)\} \subset \mathbb{R}^5$$

означает непротиворечивость геометрического смысла орбитальных элементов.

Поставлена задача нелинейного программирования. Ограничения вида неравенств означают то, что найденная орбита должна быть удалена от каждой орбиты из каталога опасных объектов как минимум на величину параметра $r_{\min}$. Орбиты, удовлетворяющие таким ограничениям будем называть *безопасными*.

1.4. Решение поставленной оптимизационной задачи. Метод штрафных функций

Известные численные методы оптимизации для решения задач, в постановке которых присутсвуют функциональные ограничения, являются, как правило, методами первого или второго порядка, которые так или иначе используют функцию Лагранжа (а также её градиент и гессиан в случае методов первого и второго порядка), в состав которой будут входить функциональные ограничения. Проблемы возникают из-за того, что функциональные ограничения-неравенства (входящие в состав лагранжиана нашей задачи) имеют достаточно сложную структуру в силу устройств функций $\rho(\boldsymbol{x}, \boldsymbol{x}_j)$. Эти функции, во-первых, вычислительно затратны (процедуры описаны в п.1.1), и, во-вторых, аналитическое вычисление градиента таких функций затруднено. Причиной этому служит то, что расстояние $\rho(\boldsymbol{x}, \boldsymbol{x}_j)$

между двумя эллипсами, определённое по формуле (4), не может быть выражено в элементарных функциях. Поэтому нет возможности передавать в аргументы метода функцию, которая будет по значению оптимизационной переменной считать её градиент. В данной работе значение (4) будет в основном вычисляться по нативной процедуре дискретизации, которая является вычислительно затратной, поэтому оценка градиента такой функции является ещё более затратной процедурой. На ранних этапах работы предпринимались попытки применять градиентные методы и решать задачу напрямую в естественной постановке (с функциональными ограничениями), однако, в силу написанного выше, этот подход оказался неудачным. Была выявлена чувствительность к числу орбит опасных объектов (числу ограничений) и, как правило, такие методы переставали сходиться, когда число ограничений превышало 6 штук. Поэтому было решено свести задачу к задаче безусловной оптимизации, чтобы расширить набор возможных применяемых алгоритмов. А конкретнее, чтобы стали доступны методы нулевого порядка, не использующие градиент. Для этого применим аппарат внешних штрафных функций. Ниже кратко будет приведена теория используемого далее метода.

Введём более общее множество (называемое *допустимым*), которому должна принадлежать оптимизационная переменная с учётом ограничений-неравенств:

$$X = \{\boldsymbol{x} \in \mathbb{R}^n |\ g_j(\boldsymbol{x}, \boldsymbol{x}_j) \leqslant 0,\ j = 1, ..., m\} \cap \Pi,$$

где в нашем случае $g_j(\boldsymbol{x}, \boldsymbol{x}_j)$ – j-ое ограничение на требуемое расстояние до опасаной орбиты. Тогда задача (12) будет выглядеть следующим образом:

$$f(\boldsymbol{x}, \boldsymbol{x}_0) \longrightarrow \min_{\boldsymbol{x} \in X}. \tag{13}$$

Определение. *Внешним штрафом для произвольного множества* $X \subseteq \mathbb{R}^n$ *называется любая функция* $\psi(\boldsymbol{x})$*, удовлетворяющая условиям:* $\psi(\boldsymbol{x}) = 0$ *при* $\boldsymbol{x} \in X$ *и* $\psi(\boldsymbol{x}) > 0$ *при* $\boldsymbol{x} \in \mathbb{R}^n \setminus X$*.*

С помощью штрафа $\psi(\boldsymbol{x})$ строим однопараметрическое семейство штрафных функций общего вида. Формально можно записать:

$$P(\boldsymbol{x}, \boldsymbol{x}_0, t) = f(\boldsymbol{x}, \boldsymbol{x}_0) + t\psi(\boldsymbol{x}, \boldsymbol{x}_0), \tag{14}$$

где $t > 0$ – *коэффициент штрафа*. При увеличении t увеличивается вклад добавки к целевой функции в недопустимых точках.

Наиболее распространённым выбором штрафа является квадратичный штраф. Таким образом, можем поставить следующую задачу оптимизации:

$$P(\boldsymbol{x}, \boldsymbol{x}_0, t) = f(\boldsymbol{x}, \boldsymbol{x}_0) + \frac{t}{2}\sum_{j=1}^{m}(g_+^j(\boldsymbol{x}, \boldsymbol{x}_j))^2 \longrightarrow \min_{\boldsymbol{x}\in\Pi}, \tag{15}$$

где $(g_+^j(\boldsymbol{x}, \boldsymbol{x}_j)) = \max\left[0, r_{min} - \rho(\boldsymbol{x}, \boldsymbol{x}_j)\right],\ j = 1, ..., m$. Таким образом имеем однопараметрическую задачу условной оптимизации с ограничением простого вида $\boldsymbol{x} \in \Pi$. Как правило, наличие таких ограничений легко учитывается оптимизаторами отдельно и их наличие сохраняет все основные свойства алгоритма метода внешних штрафных функций. Сам алгоритм можно представить следущим образом:

Algorithm 1: Метод внешних штрафных функций

Инициализация: Задаём монотонно возрастающую последовательность коэффициентов штрафа $t_k \to +\infty$;

for $k \leftarrow 1$ **to** n **do**

Шаг 1. Решая задачу оптимизации $\min\limits_{\boldsymbol{x}\in\Pi} P(\boldsymbol{x}, \boldsymbol{x}_0, t_k)$, находим точку $\boldsymbol{x}_k \in \tilde{X}(t_k)$, где $\tilde{X}(t_k)$ – множество точек, доставляющих минимум функции $P(\boldsymbol{x}, \boldsymbol{x}_0, t_k)$ при фиксированном значении коэффициента штрафа t_k.

Шаг 2. Полагаем $k := k + 1$ и идём на шаг 1.

return x_n;

Это идейное описание алгоритма внешних штрафных функций. Множество всех точек из допустимого множества X, являющихся решением (13) обозначим за X_*. Имеет место следующее утверждение.

Теорема. *Пусть* $\boldsymbol{x}_*$ *– предельная точка последовательности* $\{\boldsymbol{x}_k\}$*, полученной в ходе описанного алгоритма внешних штрафных функций. Тогда* $\boldsymbol{x}_* \in X_*$ *и*

$$\lim_{k\to\infty} P(\boldsymbol{x}, \boldsymbol{x}_0, t_k) = f(\boldsymbol{x}_*, \boldsymbol{x}_0) = f^*.$$

Выше было приведено краткое и необходимое для понимания предлагаемой методики описание теории метода внешних штрафных функций. Более подробно с данной теорией можно ознакомиться, например, в [13].

На практике обычно задаётся достаточно большое значение параметра t, при дальнейшем увеличении которого алгоритм всё равно сходится к той же точке, что и до увеличения значения этого параметра. Эта точка и будет являться предельной для последовательности $\{\boldsymbol{x}_k\}$ и, как следствие, решением задачи (15). Таким образом, в силу приведённой выше теоремы, при достаточно больших значениях параметра t задача (15) «аппроксимирует» задачу (12) в исходной естественной постановке. Так как в дальнейшем для решения задачи будут применяться методы локальной оптимизации, а функция выпуклой не является, то под решением подразумевается именно локальный минимум функции (15).

Наиболее эффективным алгоритмом для решения (15) оказался симлекс-метод Нелдера–Мида [10]. Данный алгоритм будет основным для решения поставленной задачи оптимизации.

1.5. Демонстрация работы алгоритма

В Таблице 3 приведены результаты моделирования для набора опасных орбит в количесвте 14 штук и параметра безопасного расстояния

$r_{\min} = 0.5$ тыс. км. Здесь и далее используется свободно распространяемая реализация алгоритмов оптимизации из библиотеки Scipy [14]. Для решения поставленной задачи (15), как было упомянуто ранее, используется метод Нелдера–Мида. Веса всех орбитальных элементов в функции (10) равны единице. Расстояние между орбитами в ограничениях будем считать по нативной процедуре дискретизации (8) с $N = 150$. В первых двух строчках таблицы содержится информация о начальной и найденной оптимизатором орбитах соответственно.

	a, т. км	e	i, рад.	ω, рад.	Ω, рад.	$\rho(\boldsymbol{x}_0, \boldsymbol{x}_j)$, т. км	$\rho(\boldsymbol{x}^*, \boldsymbol{x}_j)$, т. км
$\boldsymbol{x_0}$	7.01	0.015	1.69	3.0	3.77	-	-
$\boldsymbol{x^*}$	7.89	0.001	1.67	2.85	3.76	-	-
$\boldsymbol{x_1}$	7.13	0.030	1.11	2.95	0.87	0.228	0.567
$\boldsymbol{x_2}$	6.83	0.040	1.57	2.99	0.18	0.224	0.972
$\boldsymbol{x_3}$	6.93	0.010	1.57	2.99	0.27	0.199	0.994
$\boldsymbol{x_4}$	6.95	0.004	1.14	2.99	0.88	0.094	0.925
$\boldsymbol{x_5}$	6.93	0.016	1.76	2.99	0.18	0.231	0.922
$\boldsymbol{x_6}$	6.81	0.011	1.12	2.99	1.06	0.226	1.004
$\boldsymbol{x_7}$	6.75	0.014	1.58	1.59	0.36	0.289	1.113
$\boldsymbol{x_8}$	6.93	0.009	1.4	2.99	0.51	0.156	0.963
$\boldsymbol{x_9}$	6.95	0.014	1.09	2.61	0.18	0.184	1.012
$\boldsymbol{x_{10}}$	7.43	0.005	1.41	1.49	0.56	0.325	0.5
$\boldsymbol{x_{11}}$	7.43	0.008	1.45	1.85	1.4	0.362	0.55
$\boldsymbol{x_{12}}$	7.13	0.003	1.26	1.87	0.86	0.066	0.762
$\boldsymbol{x_{13}}$	6.83	0.002	1.05	1.56	0.83	0.178	1.054
$\boldsymbol{x_{14}}$	8.43	0.011	1.79	1.99	0.69	1.287	0.527

Таблица 3 – Результат численного эксперимента.

Самыми информативными и интересными для нас являются последние два столбца таблицы. Первый из них содержит информацию о расстоянии от начальной орбиты космического аппарата до каждой из опасных орбит. Отчётливо видно, что практически все значения меньше, чем заданный параметр $r_{\min} = 0.5$ тыс. км, что в введённой терминологии означает, что орбита считается опасной. В последнем столбце содержится информация о расстоянии уже от найденной оптимизатором орбиты $\boldsymbol{x}^*$ до каждой из опасных орбит $\{\boldsymbol{x}_j\}_{j=1}^{14}$. Все значения не меньше, чем заданный $r_{\min}$, что

означает, что найденная орбита $\boldsymbol{x}^*$ является безопасной. В качестве начального приближения была взята орбита

$$\boldsymbol{x}^{(0)} = \arg\min_{\boldsymbol{x}\in\Pi} \sum_{j=1}^{m} \left(g_+^j(\boldsymbol{x})\right)^2, \tag{16}$$

то есть в качестве начального приближения бралась точка с минимально возможным значением штрафного слагаемого. Это промежуточная задача оптимизации, решаемая тем же алгоритмом с начальным приближением в точке $\boldsymbol{x}_0$. Как альтернативу такому начальному приближению для основной задачи можно также использовать $\boldsymbol{x}_0$. В тех или иных ситуациях одно начальное приближение может быть лучше другого. Для большей надёжности можно решать задачу дважды с двумя разными начальными приближениями и брать то решение, которое доставляет меньшее значение функции (10). Визуализация решения для данного демонстрационного примера представлена на следующем рисунке:

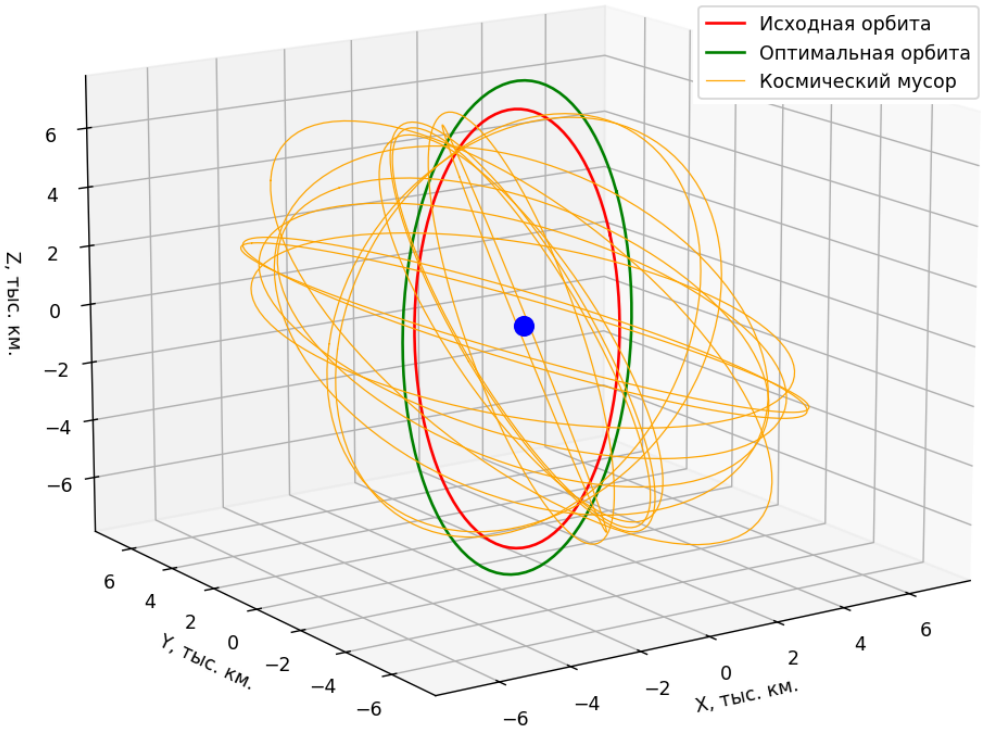

Рисунок 3 – Визуализация решения.

Алгоритм отработал за 171 итерацию, 309 раз обратился к целевой функции. На персональном компьютере с процессором AMD Ryzen 7 3750H расчёт занял 108 секунд времени. Основные вычислительные затраты на каждой итерации идут именно на вычисление расстояния между орбитами по процедуре (8). В терминах числа операций с плавающей точкой (FLOP) данная процедура имеет асимптотику $\mathcal{O}(mN^2d)$, где m, N и d – количество орбит космического мусора, количество точек дискретизации и размерность пространства соответственно. В нашем случае $d = 5$, поэтому для оценки асимптотики имеем $\mathcal{O}(mN^2)$. В случае $m \sim N$ имеем оценку $\mathcal{O}(N^3)$, что в действительности подтверждает относительную вычислительную сложность данной процедуры.

На данном же примере убедимся в отсутствии выпуклости у целевой функции задачи (15). Для этого возьмём две точки:

$$\hat{\boldsymbol{x}}_1 = (6.6, 0, 1.8, 1, 8, 1.8), \quad \hat{\boldsymbol{x}}_2 = (9.6, 0, 1.8, 1, 8, 1.8).$$

Неравенство Йенсена для целевой функции задачи (15) при фиксированной $\boldsymbol{x}_0$

$$P(\lambda\hat{\boldsymbol{x}}_1 + (1-\lambda)\hat{\boldsymbol{x}}_2) \leqslant \lambda P(\hat{\boldsymbol{x}}_1) + (1-\lambda)P(\hat{\boldsymbol{x}}_2)$$

нарушается, например, для $\lambda = \frac{7}{10}$. Поэтому целевая функция не является выпуклой на рассматриваемом множестве Π.

1.6. Границы применимости и особенности работы алгоритма

Типичное поведение работы алгоритма следующее. Как правило, на первых 50-ти итерациях происходит обнуление штрафного слагаемого и составная целевая функция сильно уменьшается, таким образом происходит условный выход на ограничения, если говорить об исходной постановке

задачи. Затем уже происходит оптимизация D-критерия и оптимизатор, не нарушая ограничений, ищет геометрически наиболее близкую к $\boldsymbol{x}_0$ орбиту. При этом были обнаружены «неудобные» для оптимизатора условия, об этом будет сказано в следующем абзаце.

В случае повышающегося числа орбит время работы алгоритма кратно увеличивается. Для тестирования алгоритма на большом количестве орбит космического мусора была создана специальная функция, которая на вход принимает одну типичную для космического мусора орбиту и возвращает любое заданное число орбит, близких к ней. К примеру, для 200 опасных орбит, сгенерированных таким образом, время работы метода составляет ≈2000 секунд. Количество итераций при этом меняется не сильно и остаётся того же порядка (может меняться как в одну, так и в другую сторону). При этом были обнаружены условно «неудобные» для алгоритма условия, которые заключаются в следующем. Если создавать плотные пучки опасных орбит на высотах, отличающихся на 1-2 тыс.км, а также задавать параметр безопасного расстояния $r_{\min} \geqslant 0.1$ тыс.км, то алгоритм «застревает» посередине наименьшей и наибольшей высот пучков орбит опасных объектов, в таком случае алгоритм не сходится. Однако, такую проблему удалось разрешить. При достижении ∼1000 итераций алгоритм останавливается и запускается заново с новым начальным приближением с увеличенным значением большей полуоси. При этом оптимизатор, как правило, уже оказывается в ситуации, когда значение штрафного слагаемого нулевое, и происходит оптимизация D-критерия. Такая схема позволяет оптимизатору во всех подобных случаях получать ответ и достигать сходимости к локальному минимуму. Но описанная выше ситуация является искусвтенной.

Кроме того, значение параметра безопасного расстояния что в демонстрационном примере, что при тестировании алгоритма на большом числе орбит космического мусора, задавалось заведомо избыточным. Чем меньше

значение этого параметра, тем быстрее метод оказывается в допустимой области и сходится к решению. Для подобных орбит характерное значение этого параметра составляет порядка 50 км, а задаваемые для наглядности значения в демонстрационном примере и тестах в разы превышают это значение. Тема грамотного определения параметра $r_{\min}$, достаточного для обеспечения безопасности, будет рассмотрена во 2-й главе данной работы.

Таким образом, границы применимости предложенной методики определяются в основном объёмом доступной оперативной памяти. Это напрямую зависит от параметра дискретизации N орбит космического мусора и количесвта таких орбит. Время выполнения алгоритма можно сократить на 1-2 порядка, перейдя на более низкоуровневый язык программирования. А при наличии параллелизма скорость может увеличиться пропорционально количеству доступных процессорных ядер.

1.7. Случай учёта зональной гармоники J_2

Из теории малых возмущений известно, что

$$\Delta\Omega \propto (-\cos i), \quad \Delta\omega \propto (4 - 5\sin^2 i), \tag{17}$$

то есть в общем случае имеет место поворот плоскости орбиты, а также поворот самого кеплерового эллипса в поворачивающихся плоскостях. Для орбит типа «Молния» ($i \approx 63.4$) имеет место только поворот плоскости орбиты, для полярной и экваториальной орбит имеет место только изменение $\Delta\omega$. На достаточно длительных промежутках времени всевозможные геометрические места точек, в которых может находиться пассивный объект на околоземной орбите, приблизительно имеют вид, представленный на рисунках 4 и 5. Такие множества будем называть *псевдоповерхностями*. Отдаление от таких поверхностей на $r_{\min}$ обеспечит безопасность космического аппарата в достаточно долгосрочной перспективе (при заданном

наборе информации об опасных объектах).

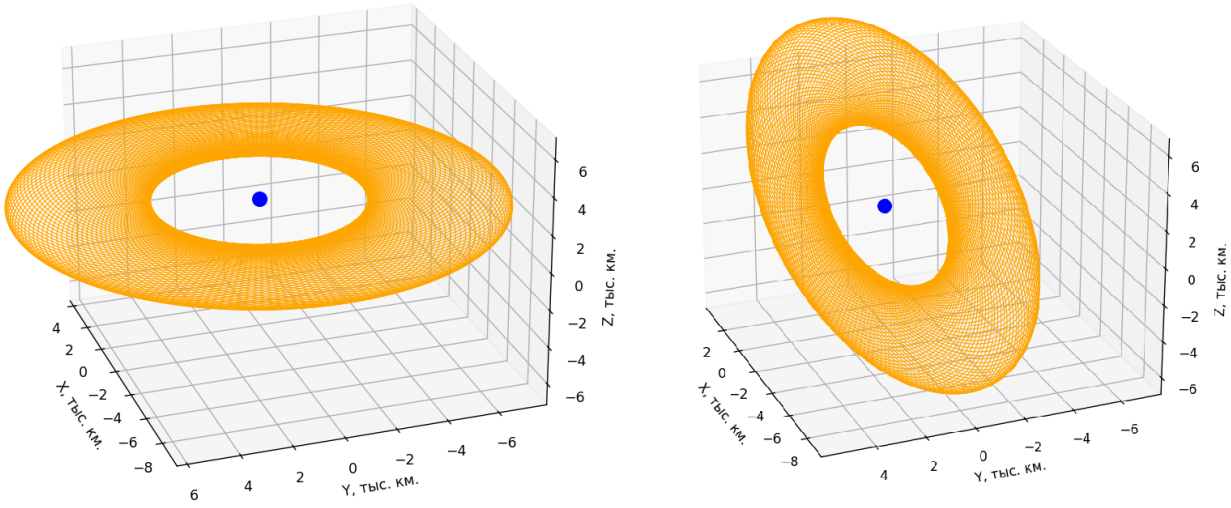

Рисунок 4 – Псевдоповерхности для наклонений $i = 0°$, $i = 90°$.

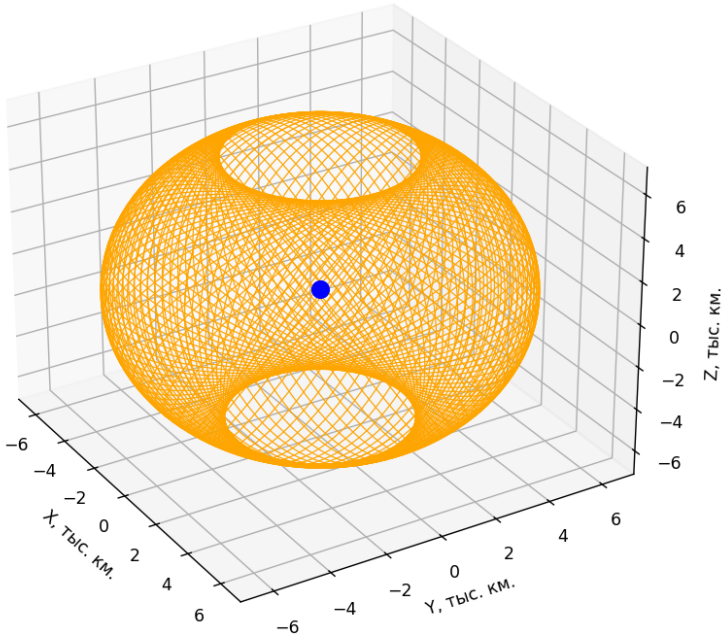

Рисунок 5 – Псевдоповерхность для наклонения $i = 63.4°$.

Програмно это можно реализовать так: строим if-else окружение, которое будет генерировать побочные орбиты (множество всех таких орбит и есть псевдоповерхности) в зависимости от наклонения i. Для полярных и экваториальных орбит ($i = \pi/2$ и $i = 0$) будут сгенерированы побочные орбиты только лишь со всевозможными значениями ω. Для орбит типа «Молния»

$(4 - 5\sin 2i = 0)$ будут сгенеированы побочные орбиты только лишь со всевозможными значениями Ω. Иначе имеем общий случай. В проводимых экспериментах количество побочных генерируемых орбит, составляющих псевдоповерхности (для каждого из значений $\omega \in [0, 2\pi], \Omega \in [0, 2\pi]$) обычно варьируется от 30 до 80. Расстояние до таких псевдоповерхностей теперь может считаться только лишь по нативной процедуре дискретизации (8).

Приведём конкретный демонстрационный пример работы алгоритма в случае учёта J_2 по предлагаемой методике. Зададим параметр безопасного расстояния $r_{\min} = 0.3$ тыс.км, плотность генерации побочных орбит по ω и Ω зададим равными 60. Имеем следующую таблицу:

	a, т. км	e	i, рад.	ω, рад.	Ω, рад.	$\rho(\boldsymbol{x}_0, \boldsymbol{x}_j)$, т. км	$\rho(\boldsymbol{x}^*, \boldsymbol{x}_j)$, т. км
$\boldsymbol{x_0}$	7.05	0.003	1.11	3.0	0.79	-	-
$\boldsymbol{x}^*$	7.61	0.003	1.01	2.92	0.80	-	-
$\boldsymbol{x_1}$	7.1	0.003	1.11	2.95	0.87	0.057	0.476
$\boldsymbol{x_2}$	7.0	0.001	1.11	1.95	0.61	0.048	0.59
$\boldsymbol{x_3}$	6.95	0.014	1.57	1.59	0.36	0.055	0.549
$\boldsymbol{x_4}$	7.22	0.001	0.0	2.59	0.43	0.158	0.365
$\boldsymbol{x_5}$	6.99	0.004	1.57	1.59	0.38	0.023	0.586
$\boldsymbol{x_6}$	7.09	0.004	0.79	1.59	0.86	0.011	0.48
$\boldsymbol{x_7}$	7.0	0.001	1.11	1.59	0.08	0.47	0.587
$\boldsymbol{x_8}$	7.3	0.003	1.57	1.59	1.11	0.224	0.3

Таблица 4 – Результат численного эксперимента с учётом J_2.

Видно, что также как и в п 1.5., оптимизатор нашёл безопасную орбиту $\boldsymbol{x}^*$, удалённую от всех псевдоповерхностей не менее чем на 0.3 тыс.км. Алгоритм успешно отработал за 140 итераций и 260 раз обратился к целевой функции. Время расчёта при этом, в сравнении с обыкновенной моделью задачи двух тел, значительно возросло. Рассчёт данного примера на том же персональном компьютере занял приблизительно 40 минут. Поведение работы алгоритма наблюдается точно такое же, как и в случае без учёта J_2, единственное отличие – сравнительно большая вычислительная сложность каждой итерации.

Отметим, что предлагаемая методика учёта зональной гармоники J_2

является обобщением уже разработанной в рамках задачи двух тел методики. Получаемые таким образом орбиты могут обеспечивать безопасность до тех пор, пока не станет известна информация о новых опасных объектах. Можно поступать иначе. А именно, можно условно искать безопасную на какое-то определённое время τ орбиту. Для этого стоит проинтегрировать уравнения движения для каждого объекта космического мусора на время τ с некоторым шагом интегрирования, получив набор точек, в которых в рассматриваемом промежутке времени может находиться каждый из объектов космического мусора. Затем для получившегося множества точек можно применить всё тот же алгоритм, что и для рассмотренных и реализованных случаев. Таким образом можно искать безопасные на время τ орбиты.

2. Методика определения параметра безопасного расстояния

В предыдущей главе в постановке задачи оптимизации фигурировал параметр $r_{\min}$, означающий по факту то расстояние, на которое в процессе оптимизации мы должны были «увести» космический аппарат ото всех орбит космического мусора из каталога. Значение этого параметра во всех демонстрационных примерах задавалось «вручную» и считалось известным. В этой главе будет предложена методика, связывающая значение этого параметра с навигационной неопределенностью положений и скоростей объектов на опасных орбитах. Предложенное далее определение данной величины может быть полезно для реального применения методики поиска безопасных орбит (описанной в главе 1) к настоящим космическим аппаратам.

2.1. Идея оценки параметра безопасного расстояния на основе неточности навигационного определения скоростей и положений опасных объектов

Будем предполагать, что возможно определить положение и скорость потенциально опасного объекта на орбите с точностью не хуже 1 км и 1 м/с соответственно. Точность определения по положениям и скоростям обозначим за σ_{pos} и σ_{vel} соответственно. Вследствие наличия неточности определения состояний опасных объектов, такие объекты через несколько витков могут отклониться от прогнозируемого положения, что может создать угрозу для космических аппаратов. Визуально предлагаемую далее идею отражает Рисунок 6. Описание данной иллюстрации следующее. Допустим объект космического мусора был определён в центральной чёрной точке. Но, как уже было сказано ранее, у такого определения есть некая

точность σ_{pos} и σ_{vel}, поэтому данный объект может находиться в любой точке из синей области (причём эти точки могут иметь разную скорость). После нескольких оборотов по орбите точки синей области «перейдут» в точки красной области. И, таким образом, объект космического мусора может отклониться от прогнозируемого положения в худшем случае на величину, обозначенную как $\Delta_{\max}$. С этой величиной и предлагается связать параметр безопасного расстояния $r_{\min}$, взяв его, например,

$$r_{\min} \geqslant \Delta_{\max}.$$

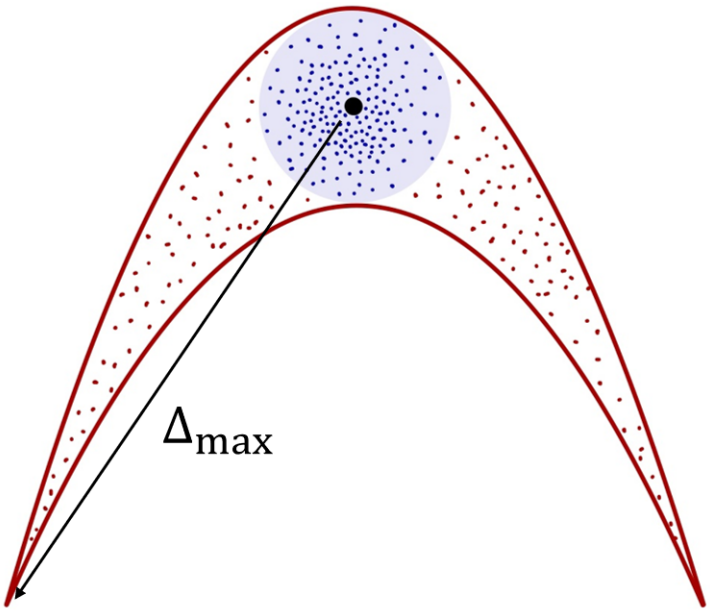

Рисунок 6 – Отклонение объекта от прогнозирумего положения.

Таким образом, удаляясь от орбит опасных объектов на такое расстояние, будет исключена возможность столкновения, связанная с тем, что невозможно абсолютно точно определить положения и скорости таких объектов.

2.2. Особенности и результаты моделирования предлагаемой методики

Для того чтобы наблюдать «переход» точек из синей области в красную (Рисунок 6), необходимо многократно решать задачу Коши с различными начальными условиями. В проводимых экспериментах будем генерировать порядка $10^5 \div 10^6$ точек фазового пространства синей области с разными положениями и скоростями согласно нормальному распределению с параметрами σ_{pos}, σ_{vel}. Соответственно, для того чтобы из синей области получить красную, нужно $10^5 \div 10^6$ раз проинтегрировать уравнения движения на нескольких орбитальных периодах. Такая процедура, даже с учётом параллелизма, может занять значительное количество времени.

Для решения данной проблемы будем использовать известный алгоритм автоматического дифференцирования. При использовании данного алгоритма предлагается однократно рассчитывать траекторию на нескольких орбитальных витках и как результат получать формулу Тейлора для аппроксимации конечного состояния аппарата в зависимости от отклонений в начальный момент времени. И, таким образом, генерируя эти различные отклонения (точки синей области), мы быстро получаем конечные состояния через несколько витков (точки красной области). Данная техника продемонстрирована, к примеру, в работе [15].

Для работы с алгоритмом автоматического дифференцирования будем использовать библиотеку `daceypy` [16]. Эта библиотека помимо реализации алгоритма автоматического дифференцирования содержит классы и функции, позволяющие работать с конкретными астродинамическими задачами.

Продемонстрируем пример работы алгоритма. Сгенерируем 10^5 точек вокруг перигея околоземной орбиты с $e = 0.5$ и минимальной высотой над поверхностью $h = 300$ км. На одном орбитальном витке имеем следующую эволюцию:

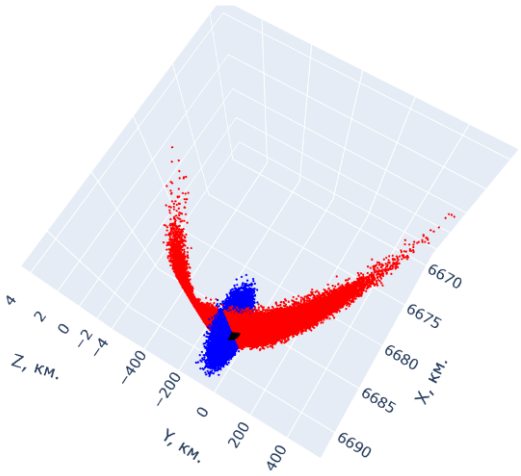

Рисунок 7 – Пример аппроксимации конечных положений космического аппарата.

В данном демонстрационном примере примере для тейлоровской аппроксимации конечных положений космического аппарата считались производные только до 3-го порядка включительно. Дальнейшее увеличение порядка тейлоровской аппроксимации практически не меняет вид красной области. В дальнейшем также будет использоваться аппроксимация именно 3-го порядка.

Теперь перейдём к строгому определению параметра безопасного расстояния на практике. Для этого возьмём низкую круговую околоземную орбиту

$$\boldsymbol{x}_{low} = (6800 \text{ км},\ 0,\ 51.6^\circ,\ 231.5^\circ,\ 227.1^\circ)^T$$

и найдём для неё множество $\mathcal{D}$ всех расстояний от чёрной точки (такими точками для круговой орбиты может быть любая её точка, а для вытянутых – перицентр) определения, вокруг которой будут генерироваться точки синей области в количестве 10^5 штук. Точности определения по положениям и скоростям зададим равными $\sigma_{pos} = 1$ км и $\sigma_{vel} = 1$ м/с соответственно. Элементы множества $\mathcal{D}_{low}$ обозначим за d. Получим такую гистограмму для плотности распределения $f(d)$ (множества $\mathcal{D}_{low}$):

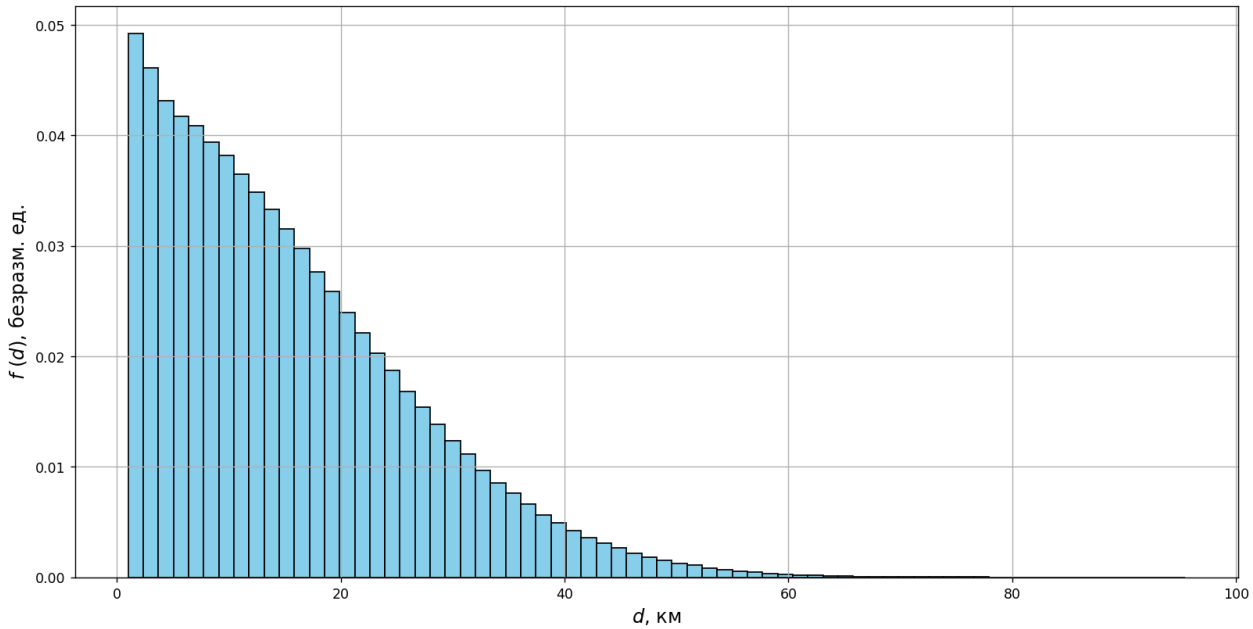

Рисунок 8 – Плотность распределения элементов множества $\mathcal{D}_{low}$ для орбиты $\boldsymbol{x}_{low}$.

Видно, что точек, удалённых на максимальное расстояние в районе 95-ти километров (обозначаемое ранее как $\Delta_{\max}$) значительно меньше, чем остальных точек выборки. Поэтому вероятность такого отклонения от прогнозируемого положения крайне мала. Вследствие этого за параметр $r_{\min}$ безопасного расстояния предлагается определить как α-квантиль распределения отклонений от прогнозируемых положений. Если $\mathcal{D}$ – такое множество, то

$$r_{\min} = Q_{\mathcal{D}}(\alpha). \tag{18}$$

В наших экспериментах по умолчанию будем полагать $\alpha = 0.9$. Иными словами, 10% наиболее удалённых точек будем считать выбросами. Выбор такого определения параметра безопасного расстояния может быть подкреплён ещё и тем, что строго говоря, нормальное распределение определено во всех точках рассматриваемого фазового пространства, из-за чего, теоретически, возможны крайне далёкие выбросы. Определение $r_{\min}$ как квантиля исключает такие ситуации.

Теперь рассмотрим, как определённый выше параметр безопасного расстояния $r_{\min}$ зависит от орбитальных элементов. Для начала посмотрим

как r_{min} зависит от большей полуоси орбиты. Выберем для примера орбиту

$$\boldsymbol{x}_1 = (6571 \text{ км},\ 0,\ 51.6^\circ,\ 251.8^\circ,\ 215.9^\circ)^T$$

и будем постепенно увеличивать значение величины большей полуоси вплоть до 7171 км, зафиксировав неизменными остальные орбитальные элементы. Навигационные неопределённости в данном примере положим $\sigma_{pos} = 1$км, $\sigma_{vel} = 10$см/с. Проделав это, можем построить следующий график:

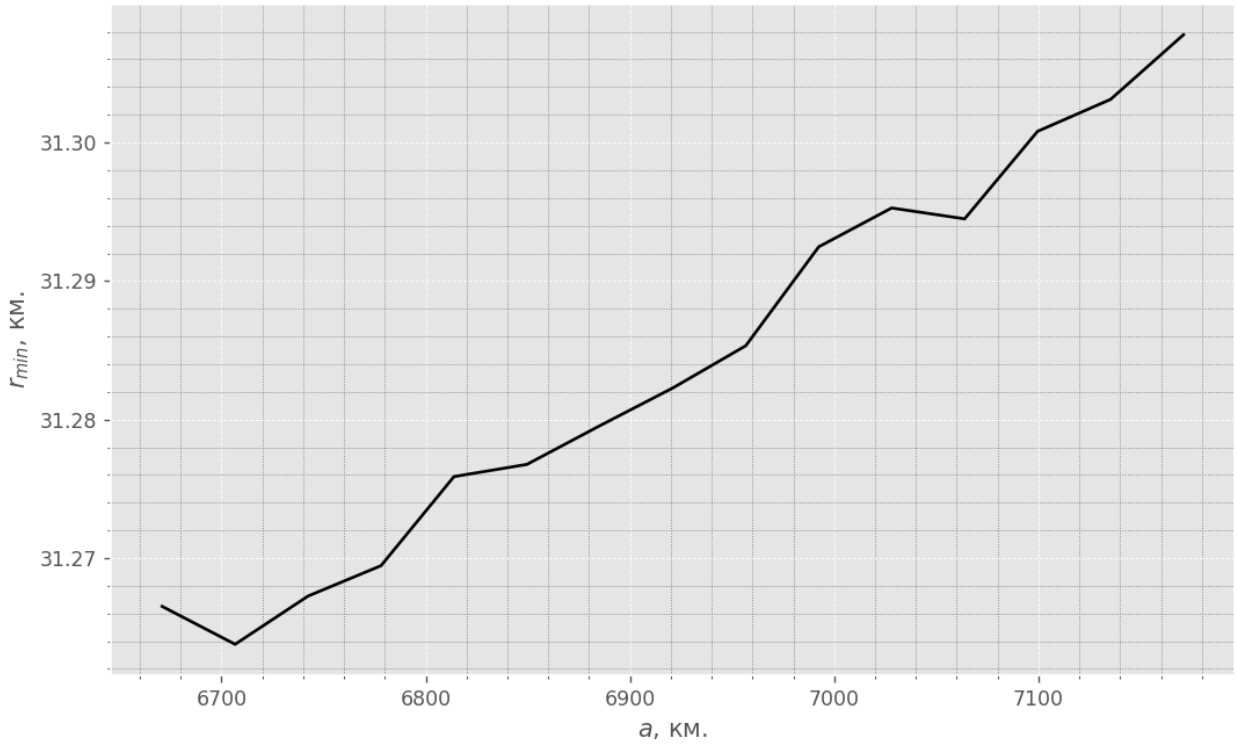

Рисунок 9 – Зависимость параметра безопасного расстония от величины большей полуоси орбиты.

В данном случае генерировалось 10^5 точек синей области отклонений в фазовом пространстве из нормального распределения с параметрами $\sigma_{pos} =$ 1 км и $\sigma_{vel} = 10$ см/с. Производные для тейлоровской аппроксимации брались после интегрирования методом РК78 [17] семейства Рунге–Кутты на одном орбитальном витке. Данный метод, обладающий высокой точностью, применяется для численного решения обыкновенных дифференциальных уравнений и обеспечивает контроль погрешности. Видим, что имеет место

тенденция увеличения значения параметра безопасного расстояния при увеличивающихся значений большей полуоси орбит.

Теперь для демонстрации выберем орбиту

$$\boldsymbol{x}_2 = (21000 \text{ км},\ 0,\ 51.6^\circ,\ 221.8^\circ,\ 115.9^\circ)^T$$

и будем постепенно увеличивать значение эксцентриситета от 0 до 0.7, зафиксировав неизменными остальные орбитальные элементы. Проделав это, можем построить следующий график:

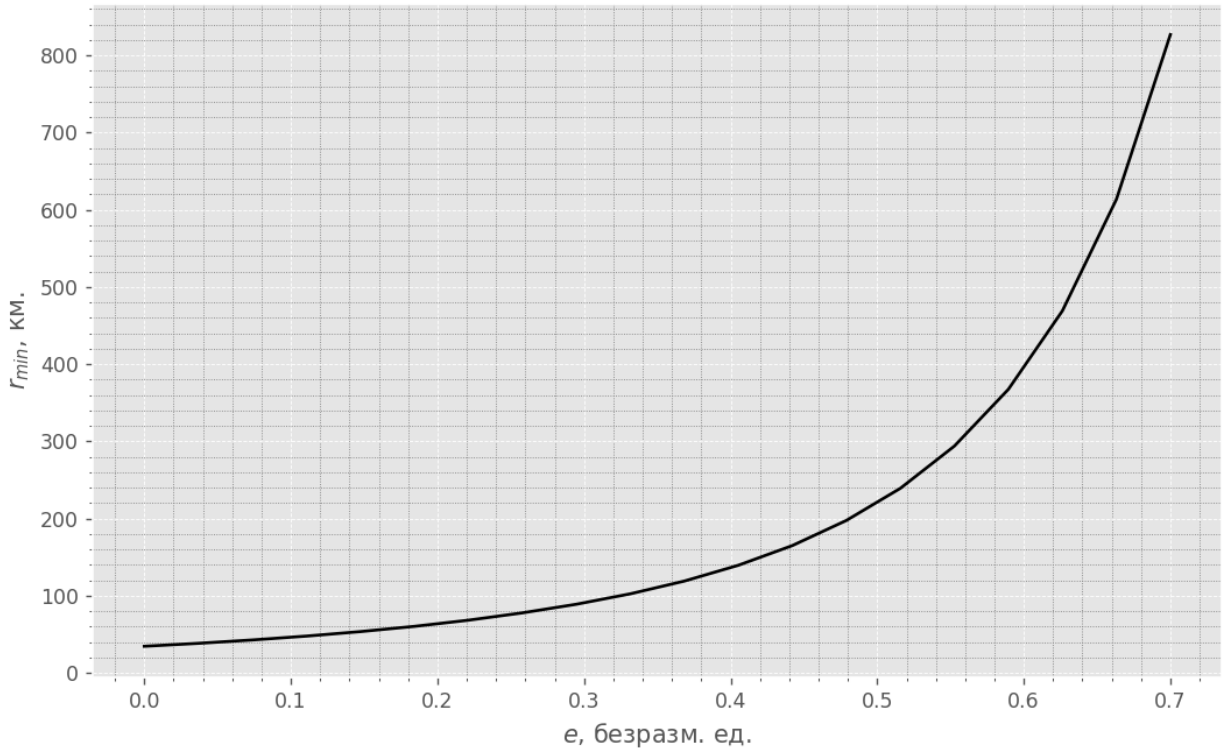

Рисунок 10 – Зависимость параметра безопасного расстония от величины эксцентриситета орбиты.

В данном случае уже видно более чёткое монотонное и нелинейное возрастание параметра $r_{\min}$ при увеличении эксцентриситета. Параметры генерации отклонений (точек синей области) в данном эксперименте точно такие же, как и для предыдущего графика.

Величина параметра безопасного расстояния $r_{\min}$ не имеет явной

зависимости от угловых орбитальных элементов (i, ω, Ω), отвечающих за ориентацию эллипса в пространстве. При аналогичных исследованиях для этих орбитальных элементов значение параметра $r_{\min}$ не имеет тенденции к явному уменьшению или увеличению, а отклонение подобных графиков от горизонталных линий объясняется стохастикой при генерации точек синей области.

2.3. Определение зависимости $r_{\min}$ от величины навигационной неопределённости. Получение расчётных формул для типичных орбит

В данном параграфе будет проведено исследование зависимсоти параметра безопасного расстояния от величины навигационных неопределённостей σ_{pos} и σ_{vel}, входящих в параметры генерации точек из синих областей по закону нормального распределения. Проведём данное исследование для солнечно-синхронной орбиты и орбиты типа «Молния».

Рассмотрим солнечно-синхронную орбиту

$$\boldsymbol{x}_{sync} = (6950 \text{ км},\ 0,\ 98.3^\circ,\ 251.8^\circ,\ 215.9^\circ)^T$$

и построим для неё график зависимости $r_{\min}(\sigma_{pos}, \sigma_{vel})$. Значения величины $r_{\min}$ для конкретной пары $(\sigma_{pos}, \sigma_{vel})$считается по процедуре, описанной в предыдущем параграфе. Параметры генерации: $5 \cdot 10^4$ точек синей области из нормального распределения, тейлоровская аппроксимация после одного орбитального оборота. Значения будут считаться в точках двумерной дискретной сетки в промежутках $(0; \sigma_{\text{pos,max}})$ и $(0; \sigma_{\text{vel,max}})$, количество значений для каждой переменной в сетке равно 50. Границы диапазонов

$$\sigma_{\text{pos,max}} = 1 \text{ км}, \quad \sigma_{\text{vel,max}} = 1 \text{ м/с}.$$

Таким образом, для круговой солнечно-синхронной орбиты имеем следующий график функции двух перменных $r_{\min}(\sigma_{pos}, \sigma_{vel})$ в виде изолиний:

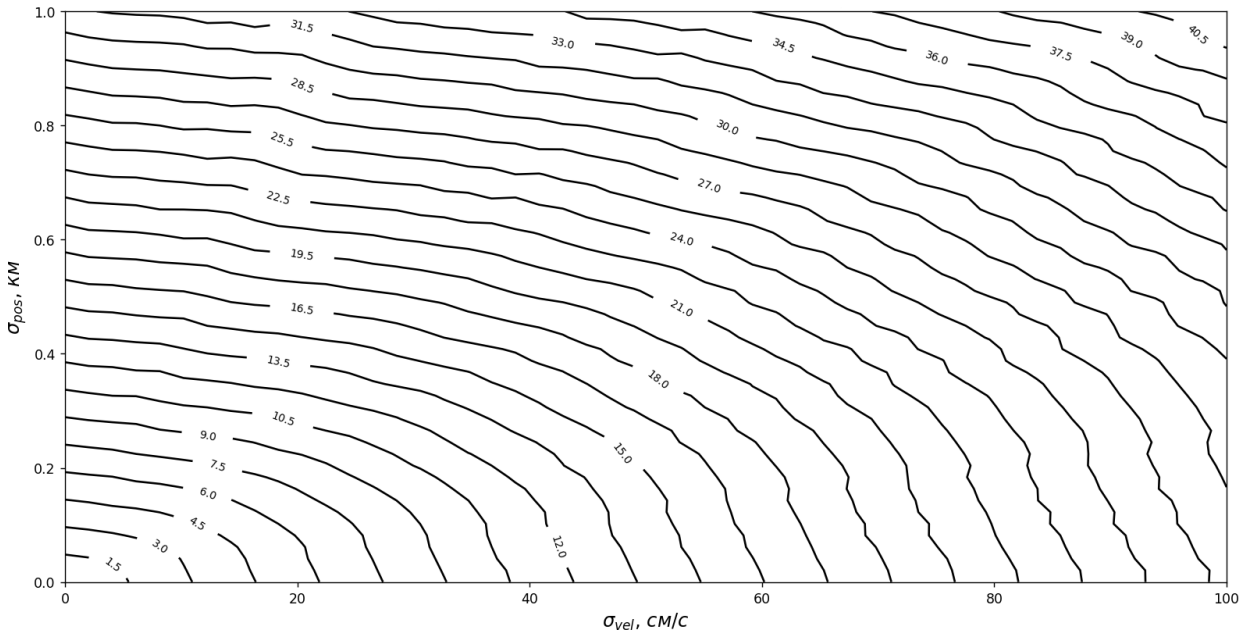

Рисунок 11 – График зависимости параметра безопасного расстояния от величины навигационной неопределённости для солнечно-синхронной орбиты.

Видим, что для данных орбит типичное значение параметра $r_{\min}$ составляет несколько десятков километров. Проделаем то же самое для орбиты тип «Молния»:

$$\boldsymbol{x}_{light} = (26600 \text{ км.},\ 0.7,\ 63.4^\circ,\ 90^\circ,\ 16^\circ)^T.$$

С теми же параметрами генерации точек отклонений имеем график, изображённый на Рисунке 12.

Обратим внимание на вид изолиний на этих графиках. Они имеют вид концентрических четвертей эллипсов. Данный факт позволяет выдвинуть предположение, что параметр безопасного расстояния может быть хорошо описан формулой:

$$r_{\min} = \sqrt[n]{\alpha_1 \sigma_{pos}^n + \alpha_2 \sigma_{vel}^n}, \tag{19}$$

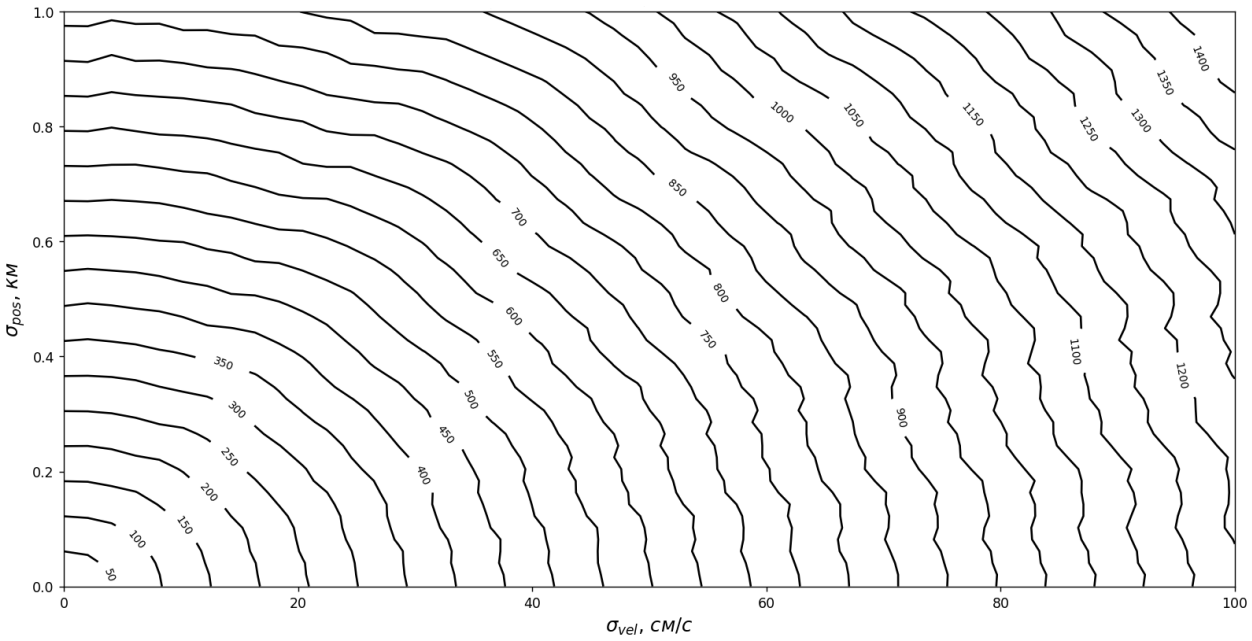

Рисунок 12 – График зависимости параметра безопасного расстояния от величины навигационной неопределённости для орбиты типа «Молния».

где $n, \alpha_1, \alpha_2 \in \mathbb{R}_+$.

Находить значения параметров n, α_1, α_2 для конкретной орбиты в формуле (19) будем в два этапа. На первом этапе зафиксируем n и поставим задачу линейной регрессии относительно α_1 и α_2:

$$\mathcal{J} = \frac{1}{M} \sum_{i=1}^{M} \left(r^n_{\min,i} - \alpha_1 \sigma^n_{\text{pos},i} - \alpha_2 \sigma^n_{\text{vel},i} \right)^2 \longrightarrow \min_{\alpha_1, \alpha_2}, \tag{20}$$

где M – количество точек в сетке, по которой были получены значения $r_{\min,i},\ i = 1, ..., M$. В данных экспериментах $M = 2500$. Задача (20) может быть переписана в векторном виде:

$$\mathcal{J} = \frac{1}{M} \sum_{i=1}^{M} \left(r^n_{\min,i} - \boldsymbol{a}_i^T \boldsymbol{\alpha} \right)^2 \longrightarrow \min_{\boldsymbol{\alpha}}, \tag{21}$$

где $\boldsymbol{a}_i = (\sigma^n_{\text{pos},i}, \sigma^n_{\text{vel},i})^T$ и $\boldsymbol{\alpha} = (\alpha_1, \alpha_2)^T$.

Для решения данной задачи имеется аналитическое выражение. Пусть $A \in \mathbb{R}^{M \times 2}$ – матрица, строки которой есть $\boldsymbol{a}_i^T$, а вектор $\boldsymbol{y} \in \mathbb{R}^M$ состоит из

компонент $r^n_{\min,i},\ i = 1, ..., M$. Тогда ответ для задачи линейной регрессии выражается формулой Маркова–Гаусса:

$$\boldsymbol{\alpha}^* = (A^T A)^{-1} A^T \boldsymbol{y}. \tag{22}$$

На втором этапе будем варьировать n и в (22) положим $\boldsymbol{\alpha}^* = \boldsymbol{\alpha}^*(n)$. Затем, подставляя $\boldsymbol{\alpha}^*(n)$ в (21) численно решаем задачу оптимизации относительно переменной n и находим

$$n^* = \arg\min_{n \geqslant 0} \mathcal{J}\left(\boldsymbol{\alpha}^*(n)\right). \tag{23}$$

Затем найденное решение задачи (23) подставляем в формулу Маркова–Гаусса и находим значения $\boldsymbol{\alpha}^{**} = \boldsymbol{\alpha}^{*}(n^*)$. Найденные значения пармаетров n^* и $\boldsymbol{\alpha}^*$ подставляются в формулу (19) и для каждой конкретной орбиты получается явная формула для оценки $r_{\min}$.

Оптимизируемый функционал в задаче (23) является функцией одной переменной n и в случае рассматриваемой выше солнечно-синхронной орбиты $\boldsymbol{x}_{sync}$ при тех же параметрах генерации отклонений имеет вид:

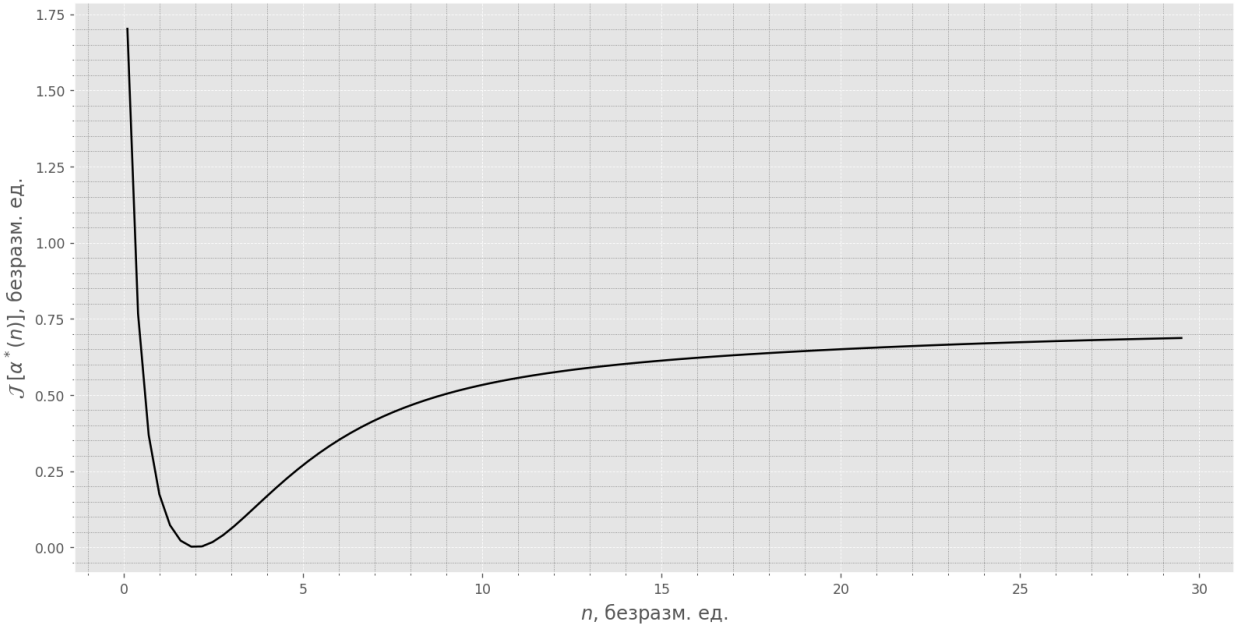

Рисунок 13 – График целевой функции задачи (23).

Решать задачу (23) в обоих случаях будем численно безградиентным методом Пауэлла [18]. Этот метод является одной из модификаций метода сопряжённых направлений. Алгоритм минимизирует функцию путем точного двунаправленного линейного поиска методом золотого сечения по каждому из специальным образом выбранных неколлинеарных направлений в пространстве $\mathbb{R}^p$, где p – размерность пространства оптимизационной переменной.

Для рассматриваемой солнечно-синхронной орбиты $\boldsymbol{x}_{sync}$ при решении (23) метод Пауэлла с начальным приближением $n^{(0)} = 2.1$ сошёлся за 2 итерации к значению $n^*_{sync} = 1.98$. В итоге, с учётом формулы Маркова-Гаусса, имеем итоговую расчётную формулу оценки $r_{\min}$ для солнечно-синхронной орбиты:

$$r^{sync}_{\min} = \left[\sqrt[1.98]{0.59(\hat{\sigma}_{\text{pos}})^{1.98} + 0.41(\hat{\sigma}_{\text{vel}})^{1.98}} \right] \cdot 41.72 \text{ км}, \tag{24}$$

где $\hat{\sigma}_{\text{pos}} = \frac{\sigma_{\text{pos}}}{\sigma_{\text{pos,max}}}$ и $\hat{\sigma}_{\text{vel}} = \frac{\sigma_{\text{vel}}}{\sigma_{\text{vel,max}}}$ – нормированные на максимум дискретной сетки значения навигационной неопределённости по положениям и скоростям соответственно.

Проделав то же самое для рассматриваемой орбиты типа «Молния», получим итоговую расчётную формулу оценки $r_{\min}$:

$$r^{light}_{\min} = \left[\sqrt[1.96]{0.32(\hat{\sigma}_{\text{pos}})^{1.96} + 0.68(\hat{\sigma}_{\text{vel}})^{1.96}} \right] \cdot 1450.03 \text{ км}. \tag{25}$$

В данном случае характер сходимости метода Пауэлла был такой же, как и в предыдущем случае солнечно-синхронной орбиты: сходимость за 2 итерации к значению $n^*_{light} = 1.96$ с начальным приближением $n^{(0)} = 2.1$.

Проверим формулу (24) на пробной точке

$$(\sigma_{\text{pos}}, \sigma_{\text{vel}}) = (0.8 \text{ км}, 20 \text{ см/сек}).$$

Значение параметра безопасного расстояния $r_{\min}$, вычисленное по процедуре из п.2.2: 25.68 км. Значение параметра безопасного расстояния $r_{\min}$, вычисленное по формуле (24): 25.46 км.

Ту же самую проверку сделаем для формулы (25) на той же пробной точке. Значение параметра безопасного расстояния $r_{\min}$, вычисленное по процедуре из п.2.2: $\approx$674.83 км. Значение параметра безопасного расстояния $r_{\min}$, вычисленное по формуле (24): $\approx$648.74 км.

Получены достаточно точные и удобные формулы для оценки параметра $r_{\min}$ для типичных орбит. Таким образом, зная к какому типу принадлежит орбита опасного объекта космичского мусора, можно получить значение параметра $r_{\min}$ для оптимизационной задачи, постановке и решению которой была посвящена глава 1. Такое часто встречается на практике, ведь, как правило, объекты космического мусора сосредоточены в основном вблизи некоторых типичных орбит.

Примечательна «близость» степени n к числу 2 в обоих случаях. Вероятно, в этом есть глубинный физический смысл, определяемый некой симметрией уравнений движения. Так как формулы (24) и (25) являются оценочными, то впоне уместно использовать

$$\begin{aligned} r_{\min}^{\text{sync}} &\approx \left[\sqrt{0.6(\hat{\sigma}_{\text{pos}})^2 + 0.4(\hat{\sigma}_{\text{vel}})^2}\right] \cdot 42 \text{ км}, \\ r_{\min}^{\text{light}} &\approx \left[\sqrt{0.3(\hat{\sigma}_{\text{pos}})^2 + 0.7(\hat{\sigma}_{\text{vel}})^2}\right] \cdot 1450 \text{ км}. \end{aligned} \tag{26}$$

3. Обзор-описание collisionAvoidance – реализованного программного инструмента для численного решения всех задач предлагаемой методики

Данная глава будет посвящена обзору-описанию реализованного на языке Python программного инструмента для численного решения задач всех разделов предлагаемой методики поиска безопасных орбит для избегания столкновения с объектами в околоземном пространстве.

Программный инструмент состоит из двух небольших вспомогательных тулбоксов `orbital_toolbox` и `da_toolbox`, а также двух главных модулей с численной оптимизацией. В параграфах данной главы будут приведены описания каждого тулбокса и главных модулей.

3.1. Пакет `orbital_toolbox`

Данный пакет состоит из 3-х основных модулей, остальные модули являются техническими.

1. Модуль `orbital_math.py` данного пакета предоставляет набор функций для работы с орбитальными элементами и расчетами, связанными с орбитами космических объектов. Ниже приводится описание главных функций и импортируемых библиотек.

Для работы с массивами и матрицами используется библиотека `numpy`, которая также содержит широкий набор математических функций. Для визуализации данных используются `plotly.graph_objects` и `matplotlib.pyplot`. Для работы с метриками расстояния используется `scipy.spatial.distance`, а для нахождения корней уравнения и решения смежных задач оптимизации используется `scipy.optimize`. Для численного

интегрирования используется `scipy.integrate`.

Функция `weighted_norm_sincos` является реализацией D-критерия (10), а функция `natural_norm` – нормы (11). Обе эти функции могут быть использованы в качестве целевой в оптимизации, находящейся в главных модулях.

Функция `elem2xyz` преобразует орбитальные элементы в декартовы координаты для заданной истинной аномалии, а `get_orbit` восстанавливает всю эллиптическую орбиту по заданной сетке истинных аномалий, которая может быть задана как `numpy.linspace(0, 2*numpy.pi, N)`, где `N` – параметр дискретизации. Используя `get_orbit`, функция `orbit_distance` вычисляет расстояние между двумя эллиптическими орбитами по процедуре дискретизации (8). Функция же `orbit_distance_polynom` вычисляет расстояние между двумя эллиптическими орбитами методом последовательной дихотомии, описанном в п.1.1, который основан на поиске всех корней ей полинома (5) на отрезке $[0, 2\pi]$.

Функция `orbit_generation` служит для генерации каталога пучков орбит космического мусора для задания ограничений, а также тестирования алгоритмов в основных модулях с оптимизацией. Получая на вход одну орбиту-набор пяти кеплеровых элементов, она генерирует несколько орбит, немного изменяя исходные орбитальные элементы поданной на вход орбиты. Таким образом, можно формировать целые группы орбит космического мусора, находящихся вблизи характерных для этого явления траекторий.

Функция `plot_orbits` является основной для визуализации решений оптизационных задач поиска безопасных орбит. Пример визуализации можно увидеть в п.1.5. Функция `plot_orbits_plotly` создаёт интерактивный 3D-график для визуализации орбит с использованием библиотеки `plotly`.

2. Модуль `orbital_generation.py` предназначен для решения задач с учётом зональной гармоники J_2. Здесь предоставлены все функции для реализации предложенной в п 1.7 методики, основанной на представлении

множества точек, в которых может находиться космический аппарат, в виде псевдоповерхностей. Важно отметить, что в данной методике не предполагается интегрирование уравнений движения, что является преимущесвом с вычислительной точки зрения. Ниже будут приведены краткие описания ключевых функций модуля (не включены описания технических функций, а также функций, дублирующихся с модуля `orbital_math.py`).

Функция `producer` на основе информации о наклонении переданной в аргумент орбиты генерирует псевдоповерхность по логике, описанной в п.1.7 (if-else окружение). По умолчанию плотность генерации по долготе восходящего узла `density_Omega=80`, а плотность генерации по аргументу перицентра `density_omega` линейным образом меняется от 0 до 100 в зависимости от эксцентриситета орбиты, это оптимизирует и исключает поворот околокруговых орбит в плоскости, что, по сути, не добавляет больше информации, а только усложняет вычисления.

Функция `produced_orbit_distance` вычисляет расстояние между двумя псевдоповерхностями, преобразуя две поданные орбиты сначала в сами псевдоповерхности функцией `producer`, а затем из сгенерированных орбит функцией `_get_orbit` (дублёр функции с предыдущего модуля) получается множество декартовых точек каждой псевдоповерхности, между которыми уже ищется расстояние с помощью нативной процедуры дискретизации (8). Есть возможность «заблокировать» генерацию псевдоповерхности первой передаваемой орбиты аргументом `controllable`, значение которого по умолчанию `controllable=False`. Это может быть полезно в случае, когда известно, что аппарат, для которого подбирается орбита, условно, способен поддерживать орбиту постоянной и компенсировать учёт J_2.

Остальные функции модуля являются либо технически вспомогательными, либо дублёрами с модуля `orbital_math.py`.

3. Небольшой модуль `orbital_dynamic.py` содержит функции, которые позволяют учитывать зональную гармонику J_2, используя численное

интегрирование уравнений движения. Функция `RSJ2` является функцией правых частей уравнений движения с учётом J_2. Функция `integrate_orbit`, принимая на вход начальные условия и врменной интервал, интегрирует уравнения с функцией правых частей `RSJ2`, возвращая множество точек (траекторию) космического аппарата на заданном интервале времени. Интегрирование производится методом РК45. Данный модуль может быть полезен, если есть необходимость ообеспечить безопасность аппарата на время τ (об этом было упомянуто в конце 1-й главы).

3.2. Пакет `da_toolbox`

Пакет `da_toolbox` содержит два основных модуля, которые содержат функции для определения параметра безопасного расстояния по методике, которой была посвящена глава 2. Основой пакета является использование библиотеки `daceypy` (версии 1.1.0).

1. Модуль `DARK78` является модулем с единсвтенной вынесенной функцией `DARK78.py`, которая, оперируя DA-объектами, сохраняет не только вектор состояния на каждом шаге, но и производные любого порядка (которые считаются с помощью автоматического дифферернцирования), интегрирование производится с использованием адаптивного метода РК78. Для этого в качестве начального состояния передаётся не вектор состояния (r, v), а специальный составной массив `array` из библиотеки `daceypy`, предварительно инициализируется «среда» DA-объектов с заданным числом вычисляемых производных и числом переменных.

2. Модуль `safety_search.py` содержит функции, необходимые для определения параметра безопасного расстояния $r_{\min}$ на основе учёта навигационной неопределённости объектов космического мусора.

Одной из двух основных функций модуля является функция `xf_finder`, которая определяет конечное положение, скорость объекта, а

также производные определённого порядка после заданного числа оборотов вокруг центрального тела. Итегрирование при этом производится функцией `DARK78` из предыдущего модуля `DARK78.py`.

Функция `rmin_finder`, принимая на вход определённую орбиту опасного объекта, количество орбитальных оборотов для получения аппроксимации конечных положений аппарата, а также число производных и величину навигационной неопределённости по положениями и скоростям, она возвращает значение параметра безопасного расстояния $r_{\min}$. Более подробно процедура описана в главе 2.

Все остальные функции либо являются вспомогательными, либо служат для построения графиков и диаграмм, которые можно увидеть в главе 2.

3. Также в данном пакете присутсвует Jupyter-notebook под названием `rmin.ipynb`, в котором реализовано получение явных оценочных формул для параметра безопасного расстояния. Процедура основана на решении задачи линейной регрессии и подробно описана в последнем параграфе главы 2. Передавая конкретную орбиту как набор пяти орбитальных элементов, число орбитальных витков для получения аппроксимации положений, можно получить готовую формулу для расчёта параметра безопасного расстояния. Для этого реализован интерфейс в самой последней ячейке ноутбука. При этом сразу же в выводе ячейки при использовании `IPython.display` будет появляться готовая математическая формула для расчёта $r_{\min}$.

3.3. Главные модули с оптимизацией

В модулях `penalty_method.py` и `dynamic_penalty_method.py` реализовано решение оптимизационной задачи поиска наиболее близких и безопасных орбит для избегания столкновений. В данных модулях активно используются основные функции из вышеописанных тулбоксов.

1. Модуль `penalty_method.py` предназначен для нахождения оптимальной безопасной орбиты методом штрафных функций с использованием безградиентных методов оптимизации в модели задачи двух тел, когда орбиты представляют собой замкнутые кеплеровские эллипсы.

Функция `x0_finder` ищет начальное приближение с минимальным возможным штрафным слагаемым. Оптимизируется только штрафная функция, и решение данной оптмимиационной задачи по умолчанию используется в качестве начально приближения основной задачи. На вход принимается орбита целевого аппарата, набор-каталог орбит опасных объектов и минимально разрешённое сближение $r_{\min}$. Оптимизация производится алгоритмом Нелдера–Мида.

Функция `double_stage_optimizer` на вход принимает все те же самые аргументы, что и `x0_finder`. Внутри неё на основе функции `orbit_distance` из `orbital_toolbox.orbital_math` анонимными функциями задаются ограничения на минимальное расстояние до опасных орбит с параметром $r_{\min}$. Также внутри данной функции определяется составная целевая функция (15), оптимизация которой будет производиться алгоритмом Нелдера–Мида. Начальное приближение по умолчанию берётся из `x0_finder`, что увеличивает время расчёта. Однако, есть возможность заблокировать этот первый этап оптмимзации и использовать в качестве начального приближения начальную орбиту космичсекого аппарата. Функция возвращает набор пяти орбитальных элементов, задающих новую оптимальную и безопасную орбиту космического аппарата.

В функции `main` происходит тестирование модуля, вызов функций с оптимизацией и задание начальных условий. При заданиии параметра $r_{\min}$ возможны два подхода: его можно задать вручную либо вычислить автоматически на основе учёта имеющейся навигационной неопределённости. В последнем случае используется функция `rmin_finder` из модуля `safety_search.py` пакета `da_toolbox`, которая с помощью генератора спис-

ков перебирает каждую орбиту из заданного каталога опасных объектов, и затем выбирается максимальное значение из полученного массива. При завершении оптимизации, функцией `plot_orbits` из модуля `orbital_math` пакета `orbital_toolbox` отрисовывается визуализация решения.

2. Модуль `dynamic_penalty_method.py` предназначен для нахождения оптимальной безопасной орбиты методом штрафных функций с использованием безградиентных методов оптимизации в случае учёта зональной гармоники J_2.

Данный модуль отличается от `penalty_method.py` лишь заданием ограничений (т.е. штрафной функции). Вместо функции `orbit_distance` из `orbital_toolbox.orbital_math` используется функция под названием `produced_orbit_distance` из `orbital_toolbox.orbital_generation`, которая автоматически кастует каждую орбиту из набора-каталога опасных объектов в псевдоповерхность и ищет расстояние между модифицированными множествами. Более подробно специфика данной функции описана в п.3.1.

Заключение

В ходе данной работы была предложена и реализована собственная методика поиска наиболее близкой к исходной и безопасной орбиты космическаого аппарата с целью избежать столкновения с космическими объектами с учётом их навигационной неопределённости. Удалось успешно решить поставленную задачу оптимизации методом штрафных функций с применением безградиентных методов оптимизации.

Перед постановкой самой задачи была также затронута тема поиска расстояния между двумя эллипсами. Были предложены и рассмотрены несколько подходов к поиску данного расстояния. Дальнейшей темой исследований может быть определение начального приближения для методов

локальной оптимизации, решающих данную задачу. Данный подход потенциально может в разы увеличить скорость расчёта расстояния между эллипсами без потери точности.

Немаловажным результатом данной работы является реализация предложенной методики определения параметра допустимого сближения $r_{\min}$. Это может быть полезно при применении предложенной методики поиска безопасных орбит в реальных задачах. Также удалось получить явные аналитические выражения для $r_{\min}$ для солнечно-синхронной орбиты и орбиты типа «Молния». Дальнейшей темой исследований также может быть более детальное аналитическое рассмотрение данной проблемы. Потенциалом этому служит довольно примечательный вид полученных формул.

Список использованных источников

[1] Bombardelli C., Hernando-Ayuso J. Optimal impulsive collision avoidance in low earth orbit //Journal of Guidance, Control, and Dynamics. – 2015. – V. 38. – №. 2. – P. 217-225.

[2] Gonzalo J. L., Colombo C., Di Lizia P. Analytical framework for space debris collision avoidance maneuver design //Journal of Guidance, Control, and Dynamics. – 2021. – V. 44. – №. 3. – P. 469-487.

[3] Morselli A. et al. Collision avoidance maneuver design based on multi-objective optimization //Advances in the Astronautical Sciences. – 2014. – V. 152. – №. 4. – P. 1819-1838.

[4] Armellin R. Collision avoidance maneuver optimization with a multiple-impulse convex formulation //Acta Astronautica. – 2021. – V. 186. – P. 347-362.

[5] Hernando-Ayuso J., Bombardelli C., Gonzalo J. L. Occam: Optimal computation of collision avoidance maneuvers //6th International Conference on Astrodynamics tools and techniques (ICATT). – 2016.

[6] Uriot T. et al. Spacecraft collision avoidance challenge: Design and results of a machine learning competition //Astrodynamics. – 2021. – P. 1-20.

[7] Raigoza K., Sands T. Autonomous trajectory generation comparison for de-orbiting with multiple collision avoidance //Sensors. – 2022. – V. 22. – №. 18. – P. 7066.

[8] The cost of space debris: In-space collisions increasingly likely. URL: `https://phys.org/news/2020-05-space-debris-in-space-collisions-increasingly.html` (дата обращения: 10.05.2024).

[9] Kholshevnikov K. V., Vassiliev N. N. On the distance function between two Keplerian elliptic orbits //Celestial Mechanics and Dynamical Astronomy. – 1999. – V. 75. P. 75-83.

[10] Nelder J. A., Mead R. A simplex method for function minimization //The computer journal. – 1965. – V. 7(4). – P. 308-313.

[11] Southworth R. B., Hawkins G. S. Statistics of meteor streams //Smith. Contrib. Astrophys. 1963. V.7. P.261-285.

[12] Kholshevnikov K. V., Vassiliev N. N. Natural metrics in the spaces of elliptic orbits //Celestial Mechanics and Dynamical Astronomy. – 2004. – V. 89. – P. 119-125.

[13] Жадан В. Г. Методы оптимизации. Ч. 2. Численные алгоритмы. § 6.1 //М.: МФТИ. – 2015.

[14] SciPy. Официальный сайт программной библиотеки SciPy [Электронный ресурс]. URL: `https://scipy.org/` (дата обращения: 27.02.2024).

[15] Di Lizia P., Armellin R., Zazzera F. B. and Berz M. High order expansion of the solution of two-point boundary value problems using differential algebra: applications to spacecraft dynamics // IRIS: Catalogo Pubblicazioni POLIMI [Электронный портал]. [2008]. URL: `https://re.public.polimi.it/handle/11311/545747` (дата обращения: 02.05.2024).

[16] Python wrapper of DACE, the Differential Algebra Computational Toolbox // GitHub : [Электронная платформа]. URL: `https://github.com/giovannipurpura/daceypy` (дата обращения: 27.02.2024).

[17] Fehlberg E. Classical fifth-, sixth-, seventh-, and eighth-order Runge-Kutta formulas with stepsize control //NASA Tech Rep R-287, NASA. – 1968. – P. 15.

[18] Powell M. J. D. An efficient method for finding the minimum of a function of several variables without calculating derivatives //The computer journal. – 1964. – V. 7(2). – P. 155-162.