# ОГЛАВЛЕНИЕ

# ВВЕДЕНИЕ

Тяга большинства современных движителей создается за счет отбрасывания масс воды, что позволяет отнести их к реактивному типу [1]. Однако способы создания реактивной тяги могут существенно различаться. Движители, создающее тягу за счет формирования подъемной силы на крыловидных элементах обладают наилучшим КПД и как следствие наиболее популярны. К ним относятся гребные винты и их разновидности, включая гребные винты в насадках, а также крыльчатые движители.

В то же время, хорошо известен принцип создания тяги за счет другой компоненты силы – силы сопротивления. Этот способ характерен для движителей, которые далее в исследовании будем называть гребными. К ним относятся весельные движители, гребные колеса и ряд бионических движителей живых существ. Уступая гребным винтам по КПД [1], эти движители способны развивать большую тягу и быстро менять вектор ее приложения, как следствие, гребные движители могут представлять интерес для специфических условий эксплуатации. Примером может служить успешная эксплуатация гребных колес на современных речных судах и разработки для амфибийных аппаратов. При этом конструкция современных гребных колес отличается от их исторического облика целым рядом усовершенствований.

Интерес исследователей к гребным движителям подогревают примеры использования данного принципа движения в живой природе, поскольку считается, что неудачные движители неизбежно были бы отброшены в процессе естественного отбора. К числу таких примеров можно отнести движение водоплавающих птиц, и в определенной степени медуз [20].

Целью данной работы является оптимизация параметров работы гребного лопастного движителя (ГЛД) [15] с целью повышения КПД данного типа движителей. Расчет КПД производится на основании ранее разработанной методики расчета характеристик ГЛД. Оптимизация проводилась с помощью метода многомерной оптимизации. В качестве метода оптимизации выступал метод Хука-Дживса. Параметрами, по которым проводилась оптимизация были: разбег a/L, поступь J,

коэффициент скорости угла поворота лопатки и коэффициенты закона перемещения. В конечном счете необходимо подобрать с помощью методов оптимизации такие параметры, при которых КПД будет максимальным при каждом конкретном случае. Оптимизация проводилась с помощью программы, написанной на языке программирования Python.

Дипломная работа имеет следующую структуру. В первой главе дается обзор методов математической оптимизации. Во второй главе дана инженерная методика расчета КПД и определение параметров оптимизации. В третьей главе приводится подробное описание выбранного метода оптимизации и обоснование выбора. В четвертой главе – описание используемых программных продуктов при выполнении исследования. В пятой главе приведены результаты и анализ проделанной работы.

# 1 Обзор методов математической оптимизации

*Теория оптимизации* – это совокупность математических и численных методов, которые позволяют не использовать полный перебор всех решений.

*Методы оптимизации* представляют собой методы, способы создания и проектирования алгоритмов для поиска оптимального значения некоторой заданной функции. Оптимальным значением может быть как минимум, так и максимум функции, в зависимости от цели поиска оптимума.

С помощью методов оптимизации можно реализовать решение задач из различных прикладных областей и сфер. Формулировка условия конкретной задачи зависит от области данной проблемы. Одним из главных моментов в теории оптимизации является правильный переход от содержательной части (описательной) постановки задачи к формализованной (математической).

Для того чтобы реализовать переход от описательной постановки задачи оптимизации к математической необходимо произвести ряд действий:

1) *Обозначить границы объекта оптимизации.* Данные границы устанавливаются с помощью пределов и служат для отделения системы или объекта от внешней среды. При этом нужно быть уверенным в том, что разделение задачи на части не повлечет за собой слишком сильного упрощения модели по сравнению с реальным объектом.
2) *Построение математической модели системы.* Математическая модель включает в себя различные уравнения, а именно: уравнения, которые описывают различные физические процессы в рассматриваемой системе; уравнения для энергетических и материальных балансов. Также в математическую модель входят неравенства, которые помогают определить область допустимых значений (ОДЗ) для переменных. Неравенства также помогают задать границы интервалов изменения характеристик интересуемого объекта.
3) *Выбор критерия оптимизации.* Критерий оптимизации может носить один из характеров: экономический, надежностный или точностной. Характер критерия оптимизации зависит от конкретно решаемой задачи. Однако вне зависимости

от его характера решением оптимизационной задачи всегда является минимум или максимум функции. Обычно, при выборе критерия оптимизации, вводят главный критерий оптимизации и вспомогательные. Вспомогательные применяют в качестве ограничений для оптимизационной задачи.

4) *Формирование целевой функции.* Следующим этапом после выбора критерия оптимизации и постройки математической модели является составление целевой функции. Данная функция состоит из математической зависимости выбранного критерия от некоторых параметров, которые влияют на его конечное значение. Вид и характер полученной функции будет определяться решаемой оптимизационной задачей. В результате, как говорилось ранее, решением оптимизационной задачи является нахождение минимума или максимума целевой функции, то есть экстремума.
5) *Построение оптимизационного алгоритма и решение задачи.* На данном шаге необходимо выбрать или разработать самостоятельно алгоритм оптимизации для решения конкретной поставленной задачи оптимизации [7].

## 1.1 Постановка задачи оптимизации

Все оптимизационные задачи различаются между собой. Однако организация у этих задач схожая. Она состоит из следующих элементов:

- *Целевая функция* $f(x)$. Это самая важная вещь в оптимизации; необходимо знать, где находить экстремум функции. Экстремум может быть максимальным или минимальным, в зависимости от поставленной цели, например, увеличение прочности конструкции при минимальном весе. Целевая функция должна быть функцией оптимизационных переменных.

$$F(x) \in R^n \tag{1.1}$$

- *Переменные оптимизации*. Это переменные, которым мы пытаемся присвоить подходящие значения для оптимизации целевой функции.
- *Ограничения*. Существуют ограничения в виде неравенств и в виде равенств. Например:

$$g_j(x) \geq 0,\ j = \overline{1,J} \tag{1.2}$$

$$h_k(x) = 0,\ k = \overline{1,K} \tag{1.3}$$

- *Область допустимых значений.*

$$x \in D \subset R^n \tag{1.4}$$

## 1.2 Классификация задач оптимизации

Оптимизационные задачи могут подразделяться исходя из типа целевой функции, размерности вектора *x*, а также ограничений. Данные задачи имеют ряд признаков, по которым их можно классифицировать. Ниже приведены главные критерии:

- *Тип параметра.* Все оптимизационные задачи можно разделить по типу параметров. Это могут быть непрерывные оптимизационные задачи, целочисленные и дискретные.
- *Ограничения.* Если задача имеет ограничения на множестве для поиска оптимума, то задача *является задачей условной оптимизации.* В противном случае – *задачей безусловной оптимизации.* Например, если в задаче $h_k(x)$ и $g_j(x)$ являются линейными, то это – задача с линейными ограничениями. Однако сама целевая функция *f(x)* не обязательно должна быть линейной.
- *Размерность.* Если данный критерий N = 1, то эти задачи называются задачами *одномерной оптимизации*. Если $N \geq 2$ – *многомерной оптимизации.*

- *Вид функции.* Исходя из вида целевой функции, а также ограничений можно различить *задачи линейного и нелинейного программирования*.
- *Характер ограничений.* По данному критерию можно разделить оптимизацию на стохастическую, то есть случайную, и детерминированную. То есть если в ОДЗ входят случайные значения, то это, определенно, стохастическое программирование.

Далее приведен Рисунок 1.1 для лучшего понимания иерархии и системы классификации задач оптимизации.

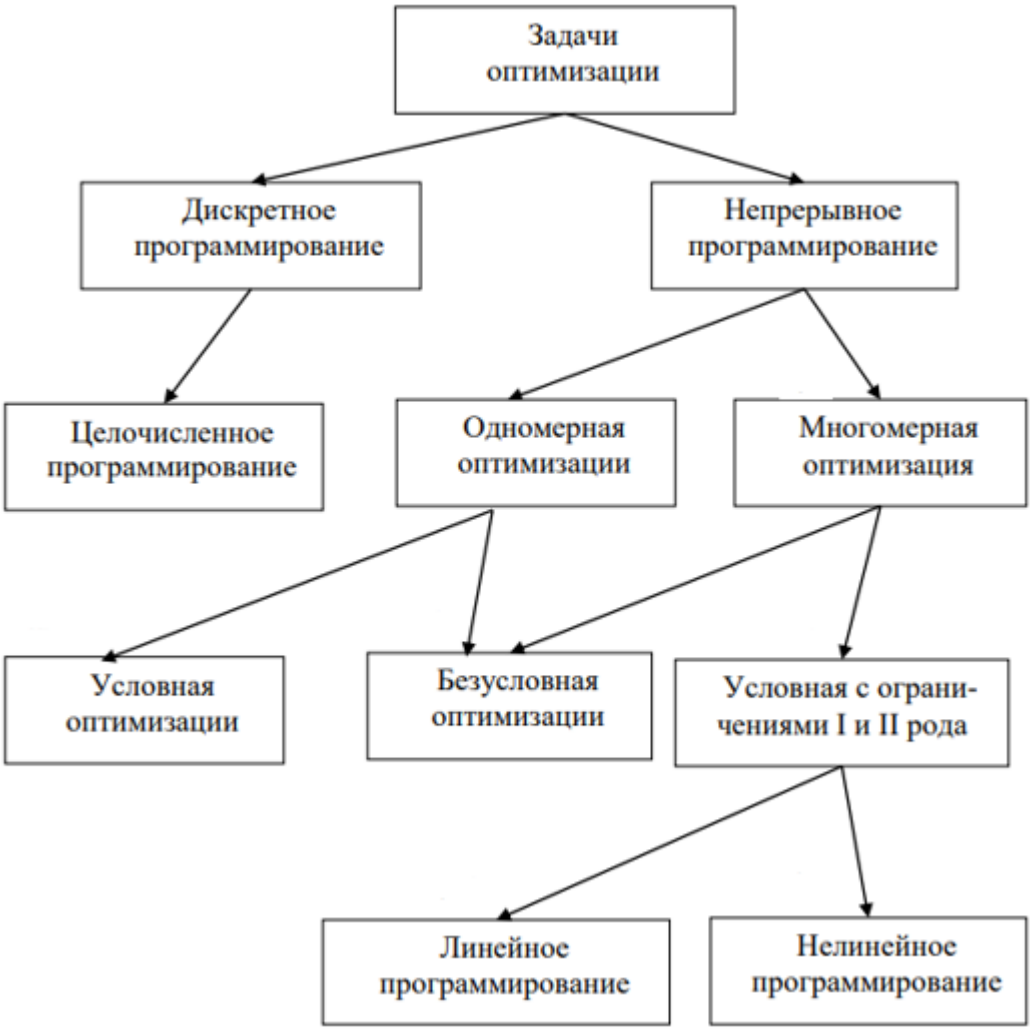


Рисунок 1.1 - Классификация задач оптимизации

*Постановка основных задач оптимизации*

Чтобы объяснить различия между постановками задач оптимизации необходимо напомнить общий вид оптимизационной задачи в формализованном виде:

$$\begin{gathered} \mathrm{F}(x) \to min \\ \text{ограничения I рода} \;\; h_k(x) = 0, \;\; k = \overline{1,K} \\ \text{ограничения II рода} \;\; g_j(x) \geq 0, \;\; j = \overline{1,J} \\ x \in D \subset R^n \end{gathered} \tag{1.5}$$

Постановка задачи безусловной одномерной оптимизации:

**min f(x),** при $\mathbf{x} \in \mathbf{R^1}$

**K = J = 0, D = $\mathbf{R^1}$**

Из названия следует, что в этой задаче ограничений нет и она имеет одномерный вектор. Целевая функция имеет произвольный вид.

Постановка задачи безусловной многомерной оптимизации:

**min F(x),** $\mathbf{x} \in \mathbf{R^n}$

**K = J = 0, D = $\mathbf{R^n}$, x ∈ [-∞,∞]** – следовательно, ограничений нет

Целевая функция имеет произвольный вид.

Постановка задачи многомерной условной оптимизации:

**min F(x)**

$\mathbf{h_k(x) = 0, k = 1, K}$

$\mathbf{g_j(x) \geq 0, j = 1, J}$

$\mathbf{x \in D, i = \overline{1, N}}$

Постановка задачи линейного программирования:

$$\mathbf{F(x) = \sum_{j=1}^{n} c_j * x_j \to min}$$

$$\mathbf{\sum_{j=1}^{n} a_{ij} * x_j, \leq b_i, i = \overline{1, m}, \overline{x} \geq 0}$$

Целевая функция является линейной функцией. Ограничения, накладываемые на оптимизационную задачу, также являются линейными.

Постановка задачи целочисленного программирования:

В данном типе задач элементы вектора X могут являться только целочисленными значениями. Как пример можно привести классическую задачу из теории оптимизации – задача о коммивояжере. Например, коммивояжер хочет объехать 5 городов так, чтобы в итоге проехать по выбранному критерию и вернуться в исходный город. Критерием могут выступать следующие величины: кратчайший маршрут, самый дешевый, самый быстрый и т.д. На практике эта задача очень часто решается в навигаторах, когда строится маршрут [2].

## 1.3 Основные характеристики алгоритмов оптимизации

После того как содержание оптимизационной задачи перенесли в математический вид, можно приступать к следующему этапу в решении задачи – выбору метода оптимизации. Для того чтобы определиться с методом оптимизации необходимо их сравнить и сопоставить по определенным характеристикам [7].

Можно выделить главные характеристики алгоритмов оптимизации:

- сходимость алгоритма
- трудоемкость вычислений
- эффективность алгоритма
- чувствительность метода к параметрам алгоритма
- устойчивость метода к погрешностям в вычислениях

В основном, оптимизационные алгоритмы основаны на итерационных методах. Данные методы имеют определенные свойства. Вот основные из них:

а) Все расчеты производятся по одной повторяющейся формуле, в которую подставляются различные значения. В большинстве случаев это означает, что каждое следующее определяемое значение будет определяться через предыдущее.

b) Чтобы начать итерационный процесс необходимо задать начальное приближение $\mathbf{x_0}$.

c) Так как итерационный процесс бесконечен, то необходимо задать условие остановки итерационного процесса. Обычно этим условием выступает формула и точность вычислений **ε** (эпсилон).

d) Одним из самых главных характеристик итерационных процессов является значение количества итераций n. Оно показывает за сколько шагов было получено решение с заданной точностью. Соответственно, что чем меньше необходимо сделать шагов, тем быстрее будет получен необходимый результат.

Рассмотрим подробнее один из важнейших показателей каждого метода оптимизации – сходимость алгоритма. Эта характеристика показывает, насколько быстро сокращается дистанция до искомого значения с точностью ε на каждой итерации. Сходимость бывает двух видов – глобальная и асимптотическая. Если при определении любой точки $x_0$ последовательность $x_n = f(x_{n-1})$ будет сходиться к точке, которая соответствует необходимым оптимизационным правилам, то такая сходимость называется **глобальной**. **Асимптотическая сходимость** представляет собой поведение ряда параметров $[x_0, x_1, x_2, \ldots, x_n]$ в окрестности предельной искомой точки $x^*$. Из этого следует, что к каждому методу и алгоритму оптимизации присваивается свой индекс эффективности O() (О большое), который называется скоростью сходимости.

«О большое» описывает насколько быстро работает алгоритм. Допустим, имеется список размера *n* и необходимо проверить список. Если использовать простой поиск, то ему нужно будет проверить каждый элемент списка и в итоге выполнить *n* операций. Время выполнения «О большое» принимает вид $O(n)$. А теперь можно попробовать проверить тот же список *n* с помощью алгоритма бинарного поиска. Для проверки списка размером n данному алгоритму потребуется $O(\log(n))$, что, соответственно, в разы быстрее простого поиска. «О большое» не сообщает за сколько секунд

выполняется алгоритм, оно сообщает сколько операций выполняется при работе алгоритма и позволяет сравнивать алгоритмы между собой таким способом [2].

## 1.4 Обзоры методов оптимизации

В данном разделе будут рассмотрены различные методы оптимизации. Все оптимизационные алгоритмы можно разделить в зависимости от критерия разделения. Например, методы оптимизации можно разделить:

- по количеству критериев оптимизации: *алгоритмы одномерной оптимизации и многомерной оптимизации.*
- по поиску экстремума: *локальные и глобальные методы оптимизации.*
- по требованиям к гладкости и наличию у целевой функции частных производных: *методы нулевого порядка или прямые методы* (необходимо только вычисление значения целевой функции в точке приближения), *методы первого порядка* (необходимо вычисление первых частных производных функции) и *методы второго порядка* (также необходимо вычисление вторых частных производных функции – гессиана рассматриваемой целевой функции).

### 1.4.1 Методы однопараметрической оптимизации

Методы однопараметрической оптимизации представляют решают самые простые задачи оптимизации, в которых нужно найти максимум или минимум функции одной переменной. Эти методы достаточно просто реализовать с помощью программного кода. Также методы одномерной оптимизации легко проверить вручную, так как все действия можно произвести письменно и вероятность ошибиться крайне мала. Кроме того, данные методы используются не только самостоятельно для решения отдельных задач, но и как часть более сложного метода многомерной оптимизации [9].

Все алгоритмы однопараметрической оптимизации могут быть разделены на 3 основные группы:

- *Алгоритмы исключения интервалов*
- *Алгоритмы точечного оценивания, использующие полиномиальную аппроксимацию*
- *Алгоритмы с использованием производных целевой функции*

Начнем рассмотрение с **методов исключения интервалов**. В данную группу методов входят такие представители, как *метод половинного деления*, *метод дихотомии*, *метод золотого сечения*, *метод равномерного поиска, метод Фибоначчи.* Ниже, на Рисунке 1.2, представлена графическая иллюстрация метода дихотомии.

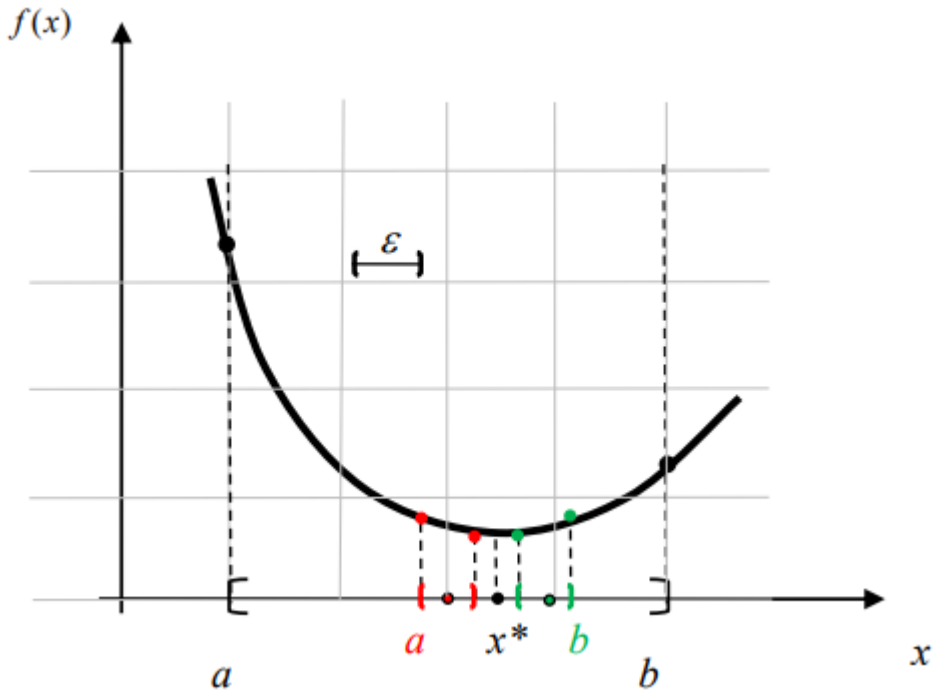


Рисунок 1.2 – Геометрическая иллюстрация метода дихотомии

Каждая группа методов предъявляет определенные требования к виду целевой функции. Например, группа методов исключения интервалов предъявляет единственное требование к функции – требование унимодальности. Данная группа методов может использоваться с непрерывными, разрывными и дискретными функциями. Далее приведен Рисунок 1.3 с видами функций.

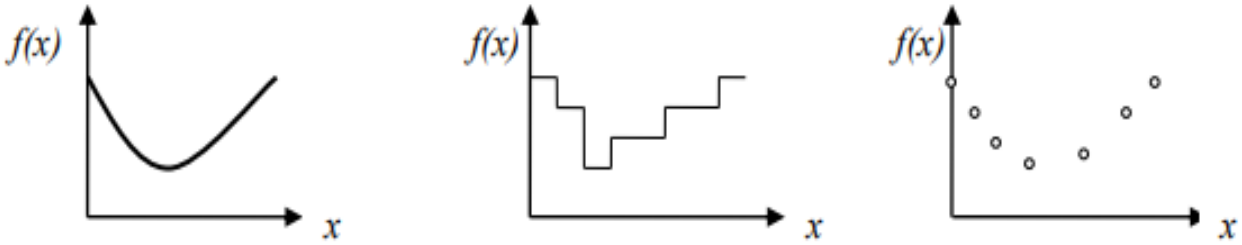


Рисунок 1.3 – Виды целевых функций в методах исключения интервалов

Все методы исключения интервалов основаны на теореме исключения интервалов. Эта теорема предписывает алгоритм последовательного исключения интервала. Данный алгоритм включает в себя 3 основных этапа:

1) Определение исходного интервала неопределенности. Точной методики для определения начального интервала неопределенности не существует и в данном случае исследователь устанавливает интервал исходя из своих догадок, опыта и логики физических процессов.
2) Сокращение первоначального интервала неопределенности с помощью выбранного алгоритма. На этом этапе и происходит поиск экстремума целевой функции.
3) Оценивание найденного экстремума на удовлетворение заданной точности ε.

Теперь рассмотрим **методы точечного оценивания**. Главным отличием от интервальных методов является то, что к функции кроме унимодальности теперь также предъявляется требование непрерывности на интервале определения экстремума.

Принцип этих оптимизационных методов основан на полиномиальной аппроксимации гладкой функции. Вычисляют значения коэффициентов многочлена для полинома по полученным результатам значения функции в отдельных конкретных точках. Далее, так как полином является легко дифференцируемой функцией, находят экстремум полученного полинома путем использования необходимых и достаточных условий экстремума.

Данная группа методов основывается на теореме Вейерштрасса, которая гласит, что любая функция, являющаяся непрерывной на интервале **[a, b]**, может быть аппроксимирована полиномом любой степени точности на данном интервале.

Чтобы повысить точность аппроксимации, а следовательно, и точность вычислений, необходимо либо увеличивать степень аппроксимации, либо уменьшать интервал аппроксимации. Но обычно, не часто применяют полиномы выше третьей степени, так как вычисление данного полинома является достаточно трудноразрешимой задачей. Поэтому, чаще всего, прибегают к уменьшению интервала аппроксимации. Ярким представителем данной группы методов является *метод квадратичной аппроксимации* или *метод Пауэлла*.

Третьей рассматриваемой группой методов являются *методы, использующие информацию о производной целевой функции*. Основными экземплярами данной группы методов являются: *метод средней точки, метод Ньютона (метод касательной), метод секущей, метод кубической аппроксимации*.

При использовании этой группы методов накладывается еще одно требование к целевой функции. Кроме того, что она должна быть унимодальна и непрерывна на интервале, целевая функция должна быть непрерывно дифференцируема. Это необходимо для того, чтобы уменьшить погрешность вычисления оптимума. Так как при использовании методов прямого поиска при вычислении значений функции f(x) появляются погрешности вычислений, на которые реагируют методы прямого поиска. Также эта погрешность при большом числе итераций возрастает и может влиять на результат. Так как функция унимодальна, и непрерывно дифференцируема, то стационарную точку $\mathbf{x}^*$ можно найти как корень уравнения $f^I(x) = 0$. Для этого и нужны данные о производной функции в точке.

Из обзора методов однопараметрической оптимизации стало ясно, что каждый метод оптимизации имеет свои положительные и отрицательные моменты. Например, методы, использующие полиномиальную аппроксимацию и данные о производных функции, намного эффективнее методов исключения интервалов. Однако этого можно достичь только при условии, что функция достаточно гладкая и непрерывная. Не каждый метод может быть применим к любой задачи. Выбор метода одномерной

оптимизации зависит от целей конкретной поставленной задачи и вида целевой функции. Цели задачи также могут накладывать ограничения на важные характеристики методов, такие как: время вычисления экстремума, точность оптимума, чувствительность метода к исходным условиям. Это сильно влияет на выбор метода оптимизации и поэтому выделяют главный критерий метода, например, скорость сходимости.

### 1.4.2 Методы многопараметрической оптимизации

Методы многомерной оптимизации используются, когда функция зависит от двух и более параметров.

Как правило, решение задач многомерной безусловной оптимизации гораздо сложнее, чем решение задач однопараметрической оптимизации, так как количество варьируемых переменных функции растет, соответственно растет количество вычислений и сложность алгоритмов. Довольно часто в методах многопараметрической оптимизации используется методы одномерной оптимизации для нахождения экстремума одного из параметров целевой функции. И эта операция может повторяться на каждой итерации алгоритма. Еще одна сложность состоит в том, что довольно сложно провести анализ поведения функции двух и более переменных [13].

Существует огромное количество методов безусловной многопараметрической оптимизации целевой функции. Как и в случае с методами однопараметрической оптимизации их можно разделить на 3 категории, в зависимости от используемой информации:

I. *Методы нулевого порядка* (*методы прямого поиска*). Данная категория методов использует при нахождении оптимума функции только значения целевой функции в точке. Они не предъявляют к целевой функции требований: регулярность и непрерывность функции не играют роли. Также не важно, чтобы функция была задана в явном виде. Это дает большое преимущество при выборе метода оптимизации, когда речь идет о сложных научных, инженерных и экономических задачах. Большинство методов прямого поиска не базируются на определенной теоретической

базе. Они разработаны исходя из интуитивных и логических соображений разработчика. Следовательно, сходимость таких методов еще мало изучена, а также сложно судить о эффективности данных методов. Однако, одним из частых способов оценки эффективности данных алгоритмов прямого поиска является вычислительный эксперимент, за которым следует анализ и сравнение результатов. Но даже такой анализ не дает стопроцентной гарантии, что один метод лучше другого.

В группу методов нулевого порядка входят такие методы как:

- *Метод покоординатного спуска (метод Гаусса-Зейделя)*. Метод покоординатного спуска является простейшим методом безусловной многомерной оптимизации. Суть метода состоит в том, что поиск экстремума целевой функции производится вдоль координатных осей параметров функции. В процессе поиска одновременно будет изменяться только одна координата, а другие будут зафиксированы. В итоге получается, что задача многопараметрической оптимизации сводится к неоднократно повторяющейся одномерной оптимизации.
- *Метод Хука-Дживса (метод конфигураций)*. Данный метод представляет собой сочетание «исследующего поиска», у которого циклически изменяются параметры и ускоряющегося поиска по образцу.
- *Метод вращающихся координат (метод Розенброка)*. Данный метод очень похож на метод Хука-Дживса. Единственное отличие состоит в принципе выбора направлений при исследующем поиске. В методе конфигураций они жестко зафиксированы и коллинеарны векторам стандартного базиса, а в методе Розенброка определение направлений происходит на ходу с помощью построения нового базиса на каждом *к*-м шаге.
- *Метод поиска по симплексу (метод Нелдера-Мида)*. Главная идея метода заключается в том, что поиск осуществляется с использованием регулярных многогранников – симплексов.
- *Метод сопряженных градиентов (метод Пауэлла)*.

II. *Методы первого порядка*. В данных методах при построении алгоритма оптимизации используется информации как о значении целевой функции в точке, так и о первой производной функции. Методы первого порядка включают в себя: метод наискорейшего спуска (Коши), метод Флетчера-Ривса, метод Поллака-Райвера.

III. *Методы второго порядка (Ньютоновские методы)*. Эти методы используют информацию о значении функции, первой производной, а также о значении второй производной при построении алгоритма поиска экстремума. К ним относятся различные модификации метода Ньютона, метод Ньютона-Рафсона, метод Марквардта.

# 2 Инженерная методика проектирования ГЛД

## 2.1 Принцип работы ГЛД

Принцип работы гребного лопастного движителя изложен в [15]. ГЛД представляет собой две лопасти прямоугольной формы, расположенные на боковых сторонах подводного аппарата. В продольном сечении ГЛД являются хорошо обтекаемым гидродинамическим профилем. В данном случае профилем лопатки является форма профиля NACA0012. Расположение ГЛД представлено на рисунке 2.1.

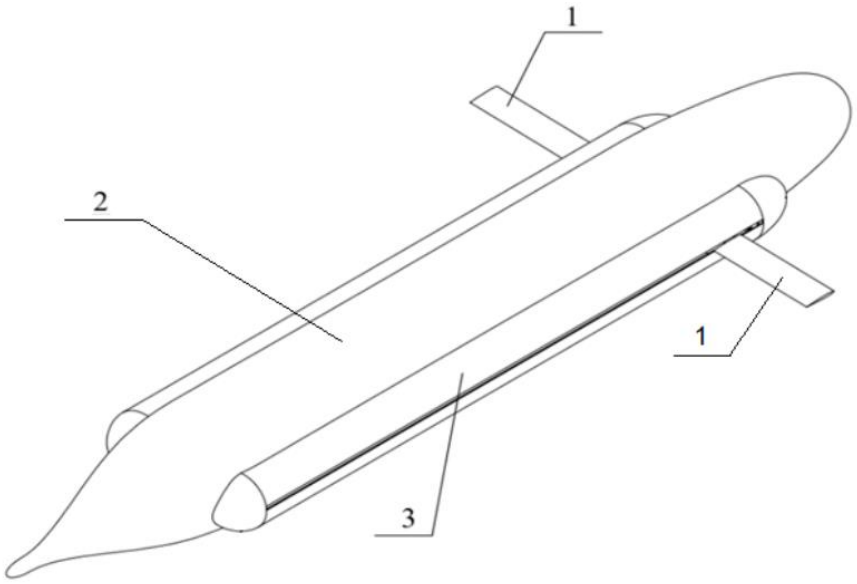


Рисунок 2.1 – Расположение ГЛД на подводном аппарате
1 – ГЛД, 2 – корпус аппарата, 3 – привод ГЛД и обтекатель

Принцип работы ГЛД состоит в отбрасывании воды назад по направлению движения аппарата, за счет движения в корму лопастей, развернутых поперек потока. Тяга, развиваемая на этом этапе, зависит периода гребка T и длины пути перемещения лопастей. Площадь лобового сопротивления лопасти, а соответственно и величина силы, действующей на нее со стороны жидкости при движении в корму, таким образом, оказывается значительно больше, чем при перемещении пластины в нос. В результате, средняя по времени сила будет направлена в сторону движения аппарата, то есть будет создаваться упор. По приближении лопастей к их крайнему кормовому

положению осуществляется их разворот на 90 градусов. В результате лопасти устанавливаются вдоль направления движения аппарата. Далее выполняется перемещение лопастей в крайнее носовое расположение. При приближении лопастей к этому положению вновь выполняется их разворот на 90 градусов, но теперь лопасти устанавливаются поперек направления движения. Далее выполняется гребок и описанный цикл повторяется. Тяга движителя в рамках представленной схемы работы создается на режиме гребка [15]. Принцип работы лопастей ГЛД изображен на рисунке 2.2.

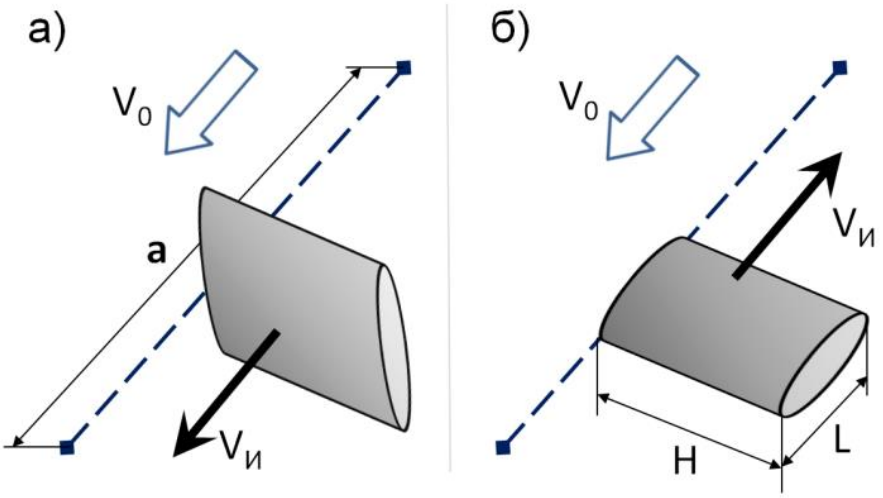


Рисунок 2.2 – Принцип работы ГЛД

а – движение в корму, б – движение в нос

В статье [12] применялся следующий закон движения:

- перемещение x (рисунок 2.3)

$$x = \frac{a/L \cdot [1 - \cos(t/T \cdot 2\pi)]^{k_2}}{2} \quad (2.1)$$

- угол поворота β (рисунок 2.4)

$$\beta = \frac{1}{2} \cdot \left(\frac{\pi}{2} + \operatorname{atan}(v_0 - x^*)\right) \quad (2.2)$$

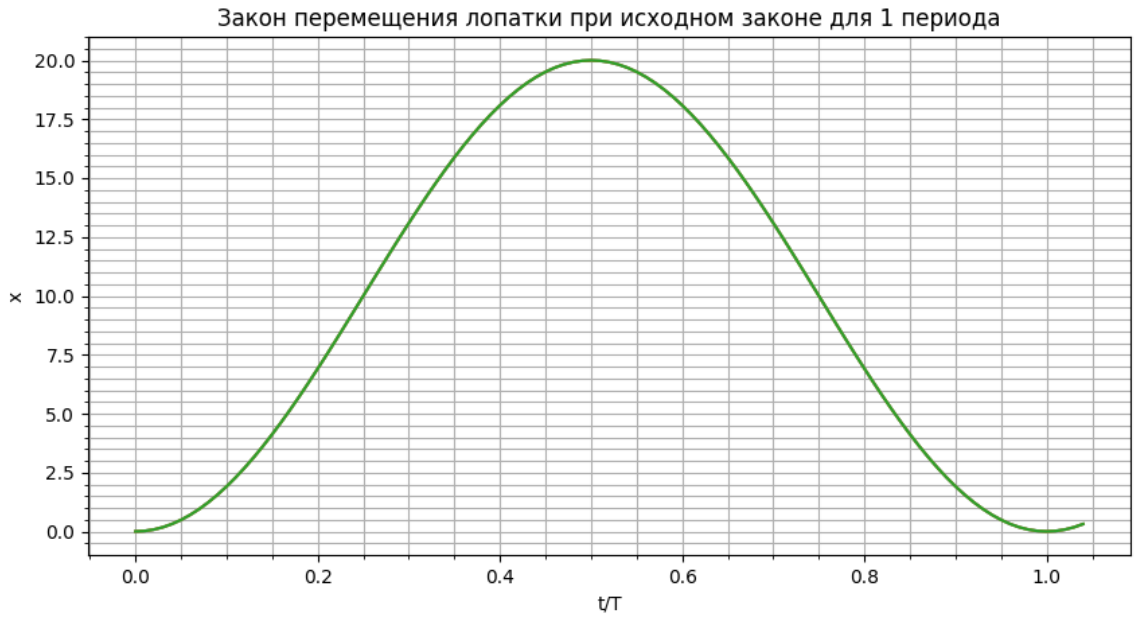


Рисунок 2.3 – Исходный закон перемещения лопатки

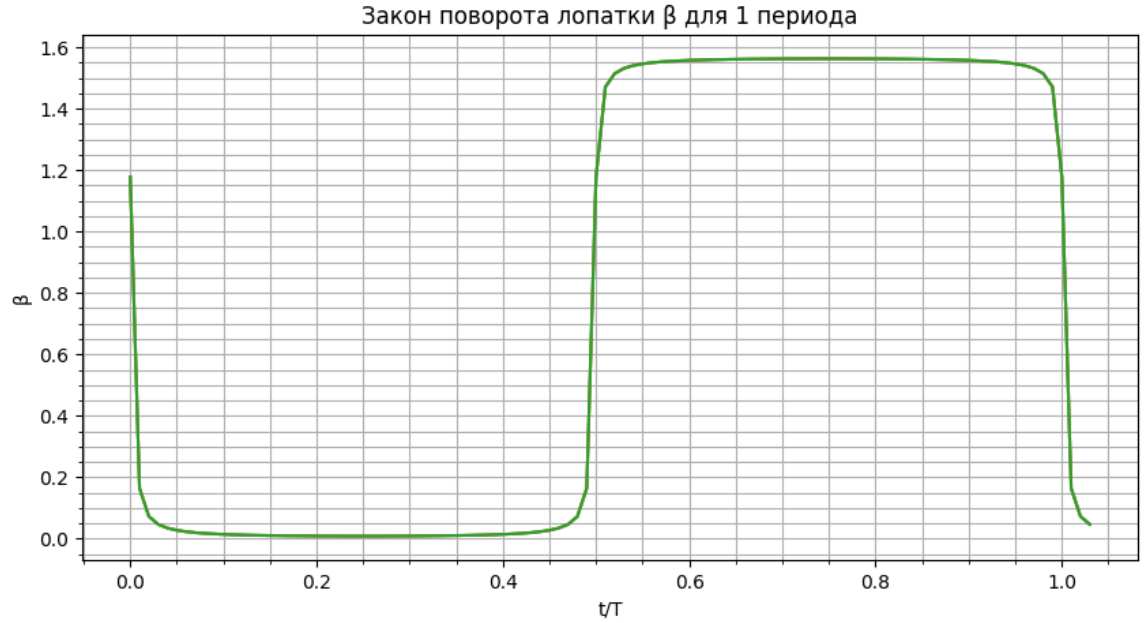


Рисунок 2.4 – Исходный закон поворота лопатки

Для увеличения КПД гребного лопастного движителя первоначальные законы движения и поворота лопасти были модифицированы следующим образом:

- закон движения лопасти был заменен на следующий:

$$x = \frac{a/L}{2}\left(1 - \frac{\left(c+\cos\left(2\pi*t/T+\frac{2c}{n}+n\right)\right)\cos n + b\sin\left(2\pi*t/T+\frac{2c}{n}+n\right)\sin n}{\sqrt{\left(c+\cos\left(2\pi*t/T+\frac{2c}{n}+n\right)\right)^2+\left(b\sin\left(2\pi*t/T+\frac{2c}{n}+n\right)\right)^2}}\right) \quad (2.3)$$

Меняя значения коэффициентов c b и n можно влиять на форму закона перемещения. Коэффициент «c» меняется в переделах -1 < c < 1 и отвечает за симметричность закона перемещения относительно T/2. Коэффициент «b» меняется в переделах 0 < b < 10 и отвечает за «полноту» синусоиды (рис. 2.5). Коэффициент n изменяет фазу закона перемещения. Предложенный закон перемещения совпадает с первоначальным при значении коэффициентов c = 0, b = 1 и n = 1.5.

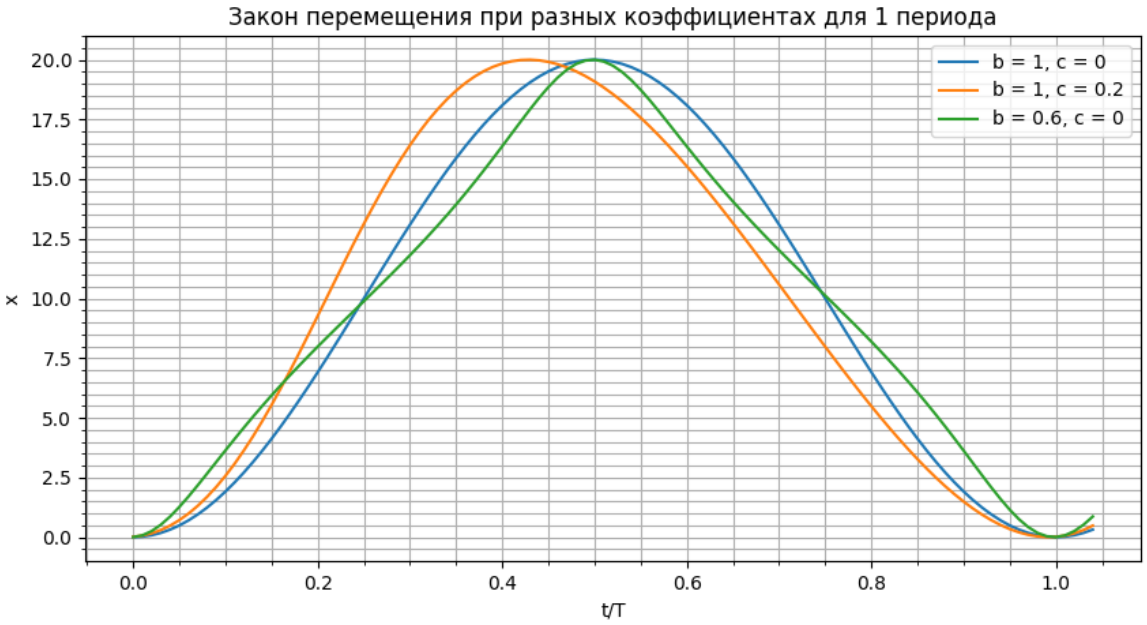


Рисунок 2.5 – Зависимость закона перемещения от коэффициентов b и c

- в закон поворота лопасти был введен коэффициент $k_\beta$ отвечающий за скорость поворота (рис. 2.6).

$$\beta = \frac{1}{2} * \left(\frac{\pi}{2} + \text{atan}\left(v_0 - k_\beta * x^*\right)\right), \quad (2.4)$$

где $v_0$ – отнормированная скорость набегающего потока, $x^*$ - первая производная по перемещению

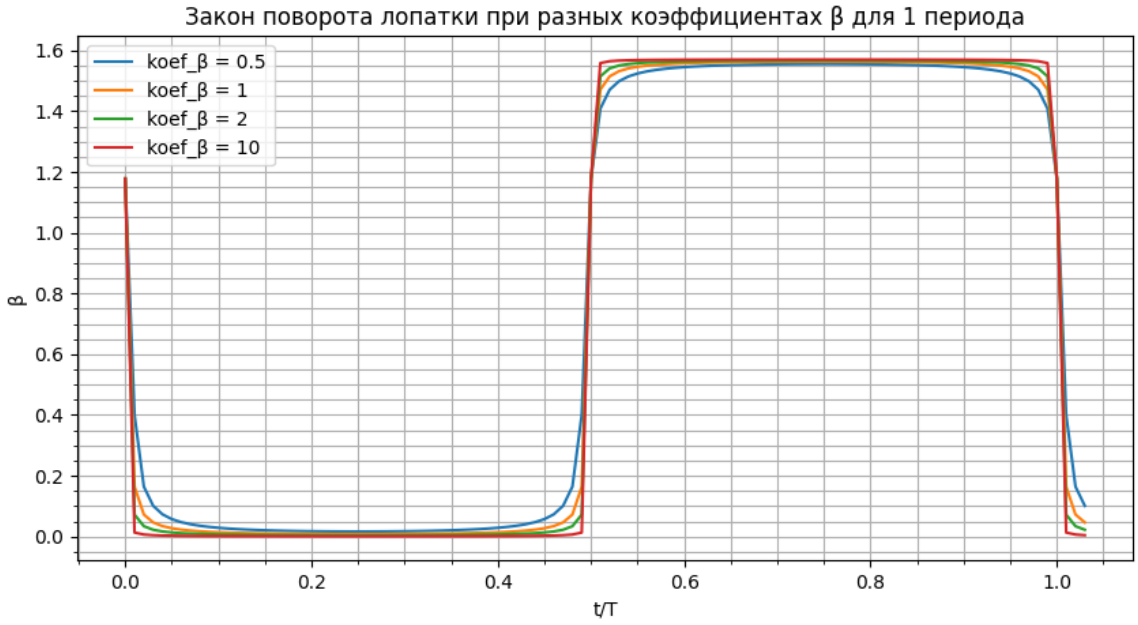


Рисунок 2.6 - Зависимость закона поворота лопатки от коэффициента β

## 2.2 Основные параметры работы гребного лопастного движителя

Схема работы ГЛД на рисунке 2.7. На этом рисунке обозначены следующие характерные линейные размеры движителя: длина хорды лопатки $L$, размах лопатки $H$, амплитуда колебаний крыла $a$. Так же в качестве параметров работы движителя можно рассматривать период колебаний крыла $T$ и скорость набегающего потока (скорость движения судна) $V_0$. В данной работе предполагается, что расположение ГЛД таково, что величину попутного потока корпуса судна можно приближенно считать равной нулю. Лопатка совершает периодические колебания в плоскости параллельной направлению набегающего потока (направлению движения судна). При этом возможно задание различных законов колебаний лопатки, включающих закон поступательного перемещения лопатки вдоль оси $0x$ $x(t)$ и закон поворота лопатки относительно оси ее вращения $\beta(t)$.

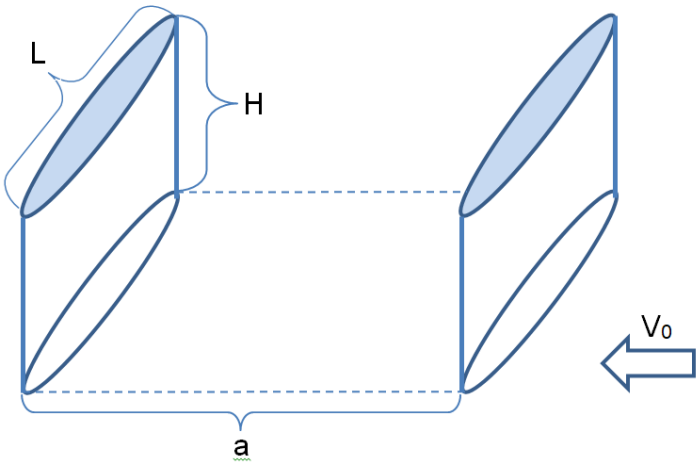


Рисунок 2.7 - Лопатка ГЛД и ее геометрические параметры.

Введенные размерные параметры позволяют получить ряд безразмерных параметров, которые будут характеризовать особенности геометрии движителя и режимов его работы. Параметры вводятся, таким образом, чтобы провести аналогию с известными безразмерными характеристиками гребных винтов и других традиционных движителей. Все линейные размеры далее будут относиться к длине хорды профиля лопатки $L$. Таким образом, ГЛД можно охарактеризовать двумя линейными параметрами:

относительным удлинением лопатки $\lambda$:

$$\lambda = \frac{H}{L} \tag{2.5}$$

относительным размахом колебаний лопатки $\bar{a}$:

$$\bar{a} = \frac{a}{L} \tag{2.6}$$

Для характеристики режима работы движителя, по аналогии с гребным винтом, вводится понятие относительной поступи движителя $J$, которое связано обратным отношением с числом Струхаля $Sh$:

$$J = \frac{\pi}{Sh} = \pi \frac{V_0 T}{L} \tag{2.7}$$

$$Sh = \frac{L}{V_0 T} \tag{2.8}$$

Расчет ведется при использовании обычного значения поступи, однако для удобства предоставления информации происходит пересчет поступи в относительную поступь по формуле (2.9).

$$J_a = \frac{J}{\pi \cdot a/L} \tag{2.9}$$

Помимо этих характеристик можно ввести дополнительное понятие относительной средней скорости движения лопатки $\bar{w}_0$, которая может быть выражена через поступь $J$ или число Струхаля:

$$\bar{w}_0 = \frac{a}{TV_0} = \frac{\pi \bar{a}}{J} \tag{2.10}$$

Основной пропульсивной характеристикой движителя является его упор, представляющий собой проекцию сил гидродинамической природы на направление движения судна $F_x$. Поскольку упор ГЛД существенно изменяется во времени, удобно ввести понятие среднего упора за один период колебаний крыла $F_{x0}$.

В качестве безразмерной пропульсивной характеристики введем понятия коэффициента нагрузки $C_T$:

$$C_T = 2\frac{F_x}{\rho LH \cdot V_0^2} \tag{2.11}$$

и коэффициента упора $K_T$:

$$K_T = \frac{F_x}{\rho aH \cdot (V_0 \bar{w}_0)^2} = \frac{J^2}{2\pi^2 \bar{a}^3} C_T \tag{2.12}$$

Для характеристики эффективности спроектированного движителя необходимо определить его КПД $\eta$. КПД движителей определяется в виде отношения полезной работы, совершенной движителем к затраченной энергии $S$. Под полезной работой понимается произведение тяги движителя на перемещение судна в единицу времени [1], затраченная энергия $S$ должна быть определена расчетным путем в рамках разрабатываемой математической модели.

$$\eta = \frac{F_{x0} \cdot V_0}{S} \tag{2.13}$$

## 2.3 Кинематика ГЛД

Для определения пропульсивных характеристик движителя с помощью метода плоских сечений перейдем к анализу его профиля. Для того, чтобы определить силы, действующие на профиле лопатки, необходимо построить соответствующие разным моментам времени треугольники скоростей. Введем понятия мгновенных скоростей поступательного перемещения профиля $V_И$ и вращения профиля $\omega_И$, которые будем определять как производные от соответствующих характеристик движения профиля:

$$V_И = \dot{x} \cdot \vec{\imath} \tag{2.14}$$

$$\omega_И = \dot{\beta} \tag{2.15}$$

где точкой сверху обозначены производные по времени.

Тогда скорость потока, набегающего на профиль, в связанной с профилем подвижной системе координат, можно представить следующим образом:

$$\vec{V} = -(V_0 + \dot{x}) \cdot \vec{\imath} \tag{2.16}$$

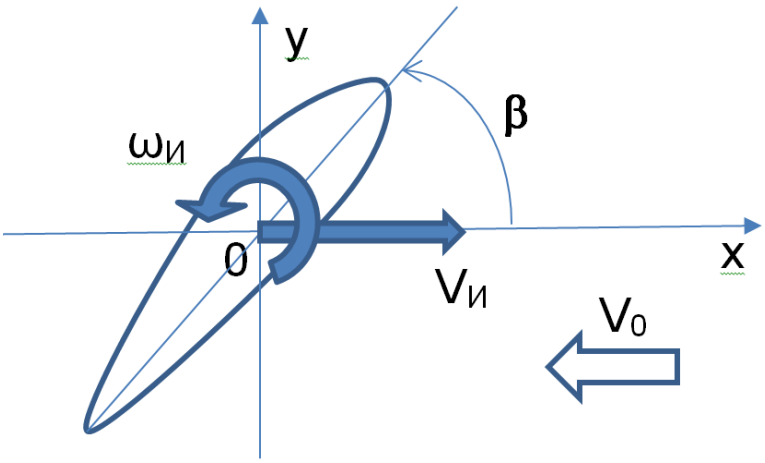


Рисунок 2.8 - Схема скоростей при обтекании профиля лопатки ГЛД

Производные по времени от x и β обезразмериваются следующим образом:

$$\begin{aligned} \dot{\bar{x}} &= \frac{\dot{x}}{V_0} \\ \ddot{\bar{x}} &= \frac{\ddot{x}L}{V_0^2} \\ \dot{\bar{\beta}} &= \frac{\dot{\beta}L}{V_0} \\ \ddot{\bar{\beta}} &= \frac{\ddot{\beta}L^2}{V_0^2} \end{aligned} \tag{2.17}$$

Кроме того, для лопатки могут существовать, так называемые индуктивные скорости, вызванные свободной вихревой пеленой, сходящей с выходящей кромки и концов крыла. Форма и интенсивность этой пелены за движущимся с переменной скоростью и со сменой направления профилем будет весьма сложной и запутанной. Если считать, что период колебания профиля невелик, то отрезок сходящей с него пелены будет постоянно срезаться при разворотах крыла. Следовательно, такой отрезок будет мал и не существенно скажется на скорости обтекания профиля. Поэтому индуктивными скоростями при построении данного алгоритма можно пренебречь.

## 2.4 Определение квазистационарных составляющих сил и момента

На профиль крыла обтекаемый потоком (2.16) будет действовать подъемная сила $Y$, перпендикулярная направлению потока и сила сопротивления $X$, направленная вдоль этого потока. Величины подъемной силы и силы сопротивления могут быть определены для профиля с известными характеристиками следующим образом:

$$Y = \rho \frac{V^2}{2} L C_Y(\beta) \tag{2.18}$$

$$X = \rho \frac{V^2}{2} L \cdot C_X(\alpha) \tag{2.19}$$

Далее можно определить квазистационарные составляющие упора и поперечной силы на лопатке ГЛД:

$$F_{CX} = -\rho \frac{V_0^2}{2} LHC_X(\beta)$$
$$F_{CY} = \rho \frac{V_0^2}{2} LHC_Y(\beta) \quad (2.20)$$

где черта сверху над производной по времени означает обезразмеривание по скорости движения судна $V_0$.

## 2.5 Определение инерционных составляющих сил и моментов

Нестационарное движение тела в жидкости, как известно [6], приводит к действию на нем дополнительных инерционных сил и моментов гидродинамической природы, которые могут быть выражены следующим образом

$$\vec{F}_{И} = -\frac{\partial \vec{B}}{\partial t} - \vec{\omega}_{И} \times \vec{B}$$
$$\vec{M}_{И} = -\frac{\partial \vec{I}}{\partial t} - \vec{\omega}_{И} \times \vec{I} - \vec{V}_{И} \times \vec{B} \quad (2.21)$$

Где $B$ – главный вектор системы мгновенных импульсов давлений, $I$ – главный момент системы мгновенных импульсов давлений относительно начала координат, $\omega_И$ – угловая скорость вращения тела относительно начала координат, $V_И$ – скорость перемещения тела (начала координат подвижной системы координат, связанной с телом).

Эти силы и моменты с достаточной для практики точностью моделируются с помощью присоединенных масс.

## 2.6 Определение характеристик ГЛД

В окончательном виде силы и моменты, действующие на лопатке ГЛД, могут быть записаны в следующем виде [3]:

$$\begin{gathered} F_X = F_{CX} - F_{ИX} - m \cdot \ddot{x} \\ F_Y = F_{CY} - F_{ИY} - m \cdot V_0 \dot{\beta} \\ M = M_C + M_И - I \cdot \ddot{\beta} \end{gathered} \tag{2.22}$$

Где $m$ – масса крыла, $I$ – момент инерции крыла относительно середины хорды.

Будем считать, что судно обладает большой инерцией и скорость его движения не зависит от пульсирующего характера сил на движителе. Тогда, применительно к судну, можно рассматривать осредненные по времени характеристики движителя. Средний упор движителя может быть рассчитан в виде следующего интеграла по времени:

$$F_{X0} = \frac{1}{T}\int_0^T F_X \, dt \tag{2.23}$$

Для расчета затрат энергии на работу движителя необходимо вычислить следующие интегралы:

$$S = \frac{1}{T}\int_0^T F_X \dot{x} \, dt + \frac{1}{T}\int_0^T M\dot{\beta} \, dt \tag{2.24}$$

На основе этих величин можно определить КПД ГЛД (2.13).

В приложении А приведена функция расчета КПД.

# 3 Описание выбранных методов оптимизации

При рассмотрении математической модели в главе 2 стало ясно, что приведенная в данной задаче целевая функция не является дифференцируемой. Поэтому невозможно использовать методы первого и второго порядков, так как они подразумевают, что функция должна быть дифференцируема.

Также, были выбраны следующие параметры оптимизации: разбег a/L, поступь J, коэффициент угла поворота $k_\beta$ и коэффициенты c и b закона перемещения.

С учетом всех рассмотренных вариантов методов оптимизации, для решения поставленной задачи оптимизации был выбран метод Хука-Дживса как для одномерной, так и для многомерной оптимизации. Этот метод относится к методам нулевого порядка, так как функция, рассчитывающая КПД ГЛД задается в неявном виде и, соответственно, невозможно получить значения первой и второй производных от этой функции.

## 3.1 Выбранный метод многопараметрической оптимизации

В качестве метода для решения задачи многомерной оптимизации был выбран метод Хука-Дживса [11]. Данный метод является наиболее эффективным среди методов прямого поиска, так как комбинирует в себе исследующий поиск, циклическое изменение переменных и ускоряющийся поиск по образцу. Также выбранный метод может использоваться, как и метод однопараметрической оптимизации, что будет необходимо в данном исследовании.

Исследующий поиск представляет собой алгоритм, с помощью которого происходит прогонка по всем координатным направлениям и в результате находит базовую точку. Чтобы провести исследующий поиск необходимо знать величину шага для каждого оптимизируемого параметра. Поиск начинается в какой-то ранее заданной точке – исходной точке. Далее к значению этой точки прибавляется шаг и происходит анализ результата функции от этой точки. Если результат целевой функции в пробной точке превышает значение целевой функции в исходной точке, то такой шаг

считается успешным. Иначе, если результат меньше, чем в исходной точке, необходимо вернуться в изначальную току и сделать шаг в противоположном направлении и проверить результат функции. По такому же принципу происходит перебор всех параметров и находится базовая точка.

Поиску по образцу выполняет соединение предыдущей базовой точки и новой, найденной с помощью исследующего поиска. Если продолжение поиска по образцу не ведет к увеличению значения целевой функции, то данная точка закрепляется как временная базовая точка, а затем снова повторяется исследующий поиск. В результате произойдет ситуация, когда поиск не даст положительных результатов. В этом случае необходимо уменьшить величину шага с помощью множителя и заново запустить исследующий поиск. Для более детального понимания работы метода Хука-Дживса ниже на рисунках 3.1-3.2 представлен алгоритм.

Достоинствами данного метода являются:

- Простая и понятная стратегия поиска
- Вычисление только значения функции
- Небольшой объем требуемой памяти
- Нет требований к форме функции
- В теории нет ограничений по количеству параметров оптимизации

Однако существуют и недостатки:

- Так как алгоритм строится на циклическом движении по координатам, то в некоторых случаях это может привести к вырождению алгоритма и отсутствию сходимости.
- Из предыдущего пункта следует, что перед оптимизацией необходимо проводить анализ функции.

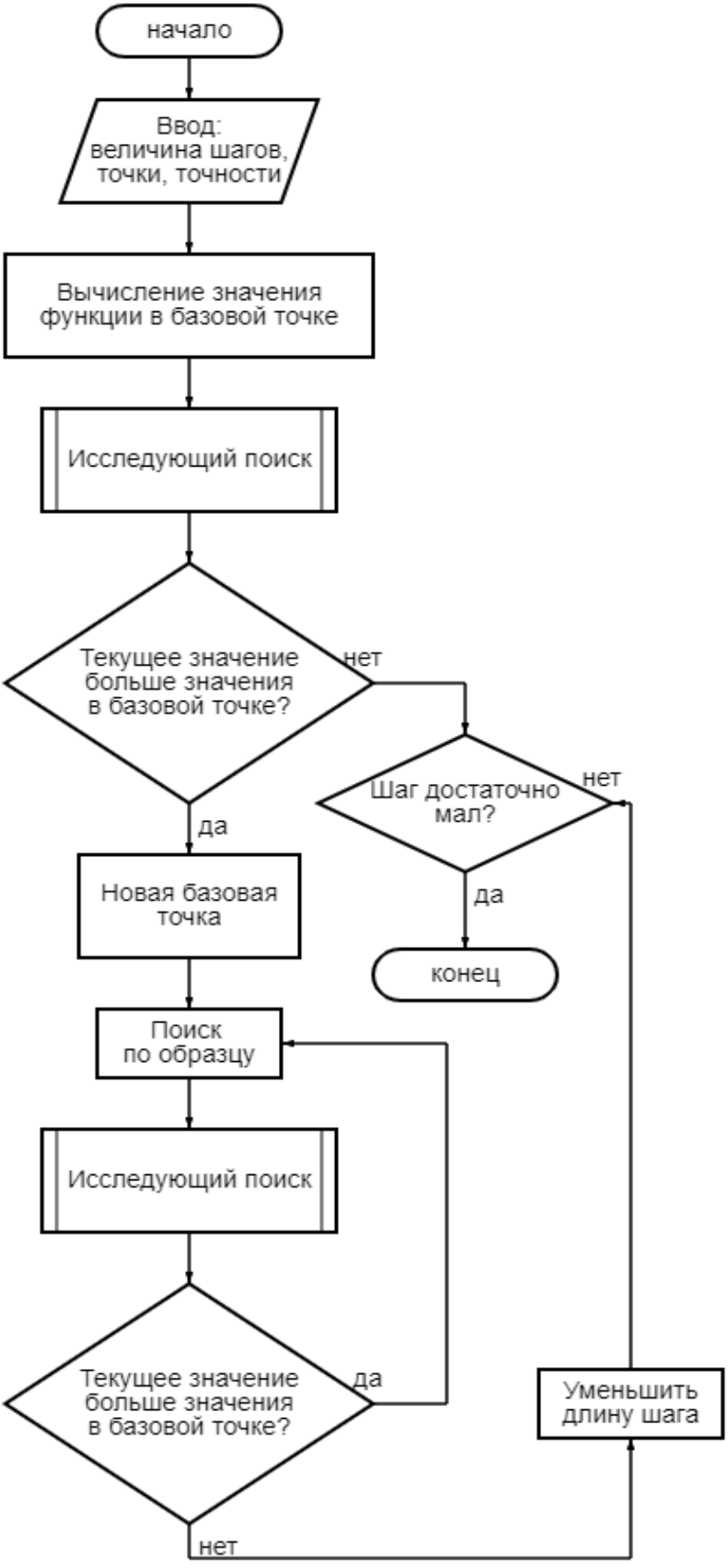


Рисунок 3.1 – Блок-схема метода Хука-Дживса

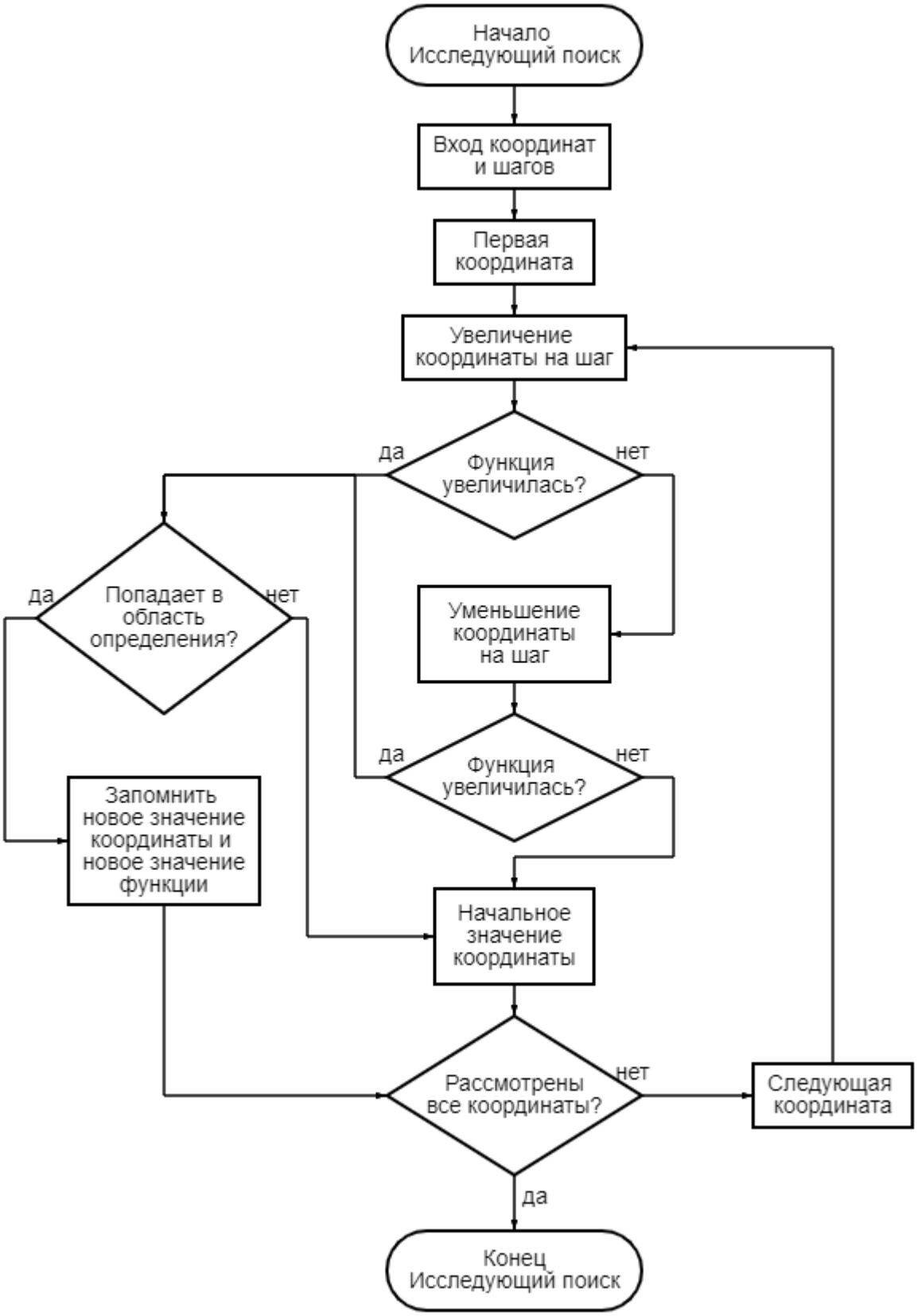


Рисунок 3.2 – Блок-схема исследующего поиска

# 4 Описание используемых программных продуктов

В данной главе будут рассмотрены различные программные продукты, с помощью которых создавалось программное обеспечение для расчета функции КПД, реализовались методы оптимизации, а также анализировались полученные результаты.

Главным инструментом для создания функции по расчету КПД и реализации методов оптимизации стал высокоуровневый язык программирования Python версии 3.9.0. Данный язык программирования является интерпретируемым [18]. Он идеально подходит для выполнения практически любой задачи, так как синтаксис предельно понятен. Из этого следует, что можно уделить больше времени на освоение и написание алгоритмов, чем на разбор синтаксиса языка. Следовательно, порог вхождения является достаточно низким. Данный язык является очень популярным, поэтому имеет большое сообщество программистов, которые его дополняют различными модулями на любую тематику – библиотеками.

Python имеет достаточно обширную стандартную библиотеку, в которой можно найти множество инструментов для реализации любой программы. Также одной из причин выбора этого языка для написания программы является то, что имеется огромное количество сторонних пакетов для реализации как научных, так и математических задач. Главными примерами этих библиотек являются: NumPy (Numeric Python), SciPy (Scientific Python), MatPlotLib.

Обычно, данные три библиотеки устанавливаются одновременно, так как NumPy расширяет функционал стандартного языка Python, а SciPy, в свою очередь, расширяет возможности NumPy. Библиотека MatPlotLib необходима для удобного и красивого вывода различного типа графиков. С помощью этого модуля можно максимально точно настраивать графики.

Библиотека NumPy ощутимо расширяет возможности работы с массивами и матрицами данных. В то же время, SciPy расширяет возможности с помощью огромного количества алгоритмов, таких как минимизация, преобразование Фурье, регрессия, вычисление интегралов, дифференцирование, решение уравнений, обработка сигналов, различный операции линейной алгебры, обработка изображений [17]. С

помощью данных установленных пакетов, Python может с легкостью заменить своим функционалом известный язык программирования MATLAB.

Также, для анализа полученных результатов использовались такие программы как Excel и онлайн-программа для построения различных графиков и их анализа – Desmos.

# 5 Анализ полученных результатов

## 5.1 Проверка работоспособности метода

Для начала проведения оптимизации поставленной задачи необходимо убедиться в правильности работы выбранного и запрограммированного метода оптимизации. Напомню, что в качестве метода многопараметрической оптимизации был выбран метод Хука-Дживса.

Чтобы убедиться в правильности работы алгоритма, необходимо протестировать его на такой целевой функции, для которой известна точка минимума и при каких значениях параметров она достигается. В качестве такой функции для оценивания эффективности работы алгоритмов, в 1960 году Ховардом Розенброком [19], была предложена функция Розенброка (5.1). Данная функция имеет минимум равный 0 в точке [1, 1]. Данная функция приведена на рисунке 5.1.

$$\mathrm{f}(x, y) = (1 - x)^2 + 100 * (y - x^2)^2 \tag{5.1}$$

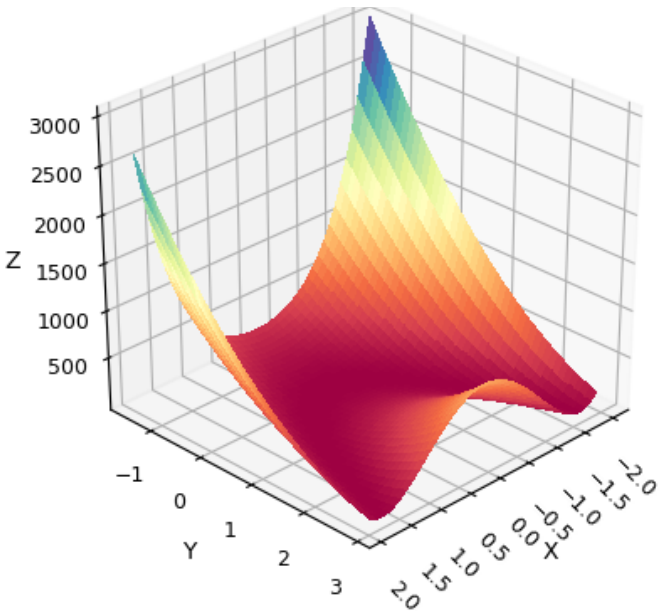


Рисунок 5.1 – График функции Розенброка

В результате оптимизации функции Розенброка, запрограммированным методом Хука-Дживса (приложение Б), с точностью оптимизации **ε** = 1e-3 были получены следующие результаты параметров: [1.0001500000050465, 1.0003499999910703]. Значение функции Розенброка при данных параметрах равно: 2.723e-07. Следовательно, метод оптимизации работает правильно и его можно использовать для решения задачи.

## 5.2 Ход решения и получение результатов

Как уже упоминалось, исходные законы движения и поворота лопатки были заменены на законы (2.3) и (2.4) соответственно. Также были уже определены параметры оптимизации в главе 3.

В начале расчетов берется диапазон разбега a/L от 0 до 100 и проводится двухпараметрическая оптимизации по поступи J и разбегу a/L при исходных коэффициентах и исходном законе движения. Были получены следующие результаты: максимальный КПД достигается при разбеге a/L = 100 и относительной поступи Ja = 1.256. КПД при данных значениях коэффициентов, a/L и Ja составляет 0.342.

Далее, для полученных значений a/L и J проводится оптимизация по коэффициенту β и коэффициентам b и c. При оптимизации были получены следующие значения:

- koef_β = 1167
- b = 0.563
- c = -0.229

При данных значениях коэффициентов КПД возрос до 0.452.

Полученные коэффициенты законов движения лопасти были также опробованы для значений разбега a/L = 2, 3, 4, 5, 10, 20, 50, 100. Для каждого их этих разбегов, путем однопараметрической оптимизации были получены значения относительных поступей, на которых наблюдается максимальный КПД при первоначальном

законе движения. В результате были выделены восемь режимов движения, результаты которых находятся в таблице 5.1:

| a/L | Ja | КПД |
|---|---|---|
| 2 | 0.499 | 0.120 |
| 3 | 0.675 | 0.154 |
| 4 | 0.835 | 0.206 |
| 5 | 0.956 | 0.252 |
| 10 | 1.160 | 0.325 |
| 20 | 1.205 | 0.331 |
| 50 | 1.241 | 0.342 |
| 100 | 1.256 | 0.349 |

Таблица 5.1 – 8 значений a/L и КПД при исходном значении коэффициентов

Результаты применения полученных коэффициентов модифицированных законов движения попытки представлены в таблице 5.2 из которых видно, что на некоторых значениях разбегов КПД становится меньше 0, что не является физически возможным.

| a/L | Ja | koef_β | c | b | КПД |
|---|---|---|---|---|---|
| 2 | 0.796 | 1167 | -0.229 | 0.563 | -0.002 |
| 3 | 1.061 | 1167 | -0.229 | 0.563 | -0.081 |
| 4 | 1.273 | 1167 | -0.229 | 0.563 | -0.211 |
| 5 | 1.401 | 1167 | -0.229 | 0.563 | -0.539 |
| 10 | 1.552 | 1167 | -0.229 | 0.563 | 0.557 |
| 20 | 1.459 | 1167 | -0.229 | 0.563 | 0.449 |
| 50 | 1.474 | 1167 | -0.229 | 0.563 | 0.445 |
| 100 | 1.490 | 1167 | -0.229 | 0.563 | 0.452 |

Таблица 5.2 – Результаты при коэффициентах для a/L = 100

Для получения более полной картины, были построены графики зависимости КПД от поступи для рассматриваемых разбегов (рис. 5.3)

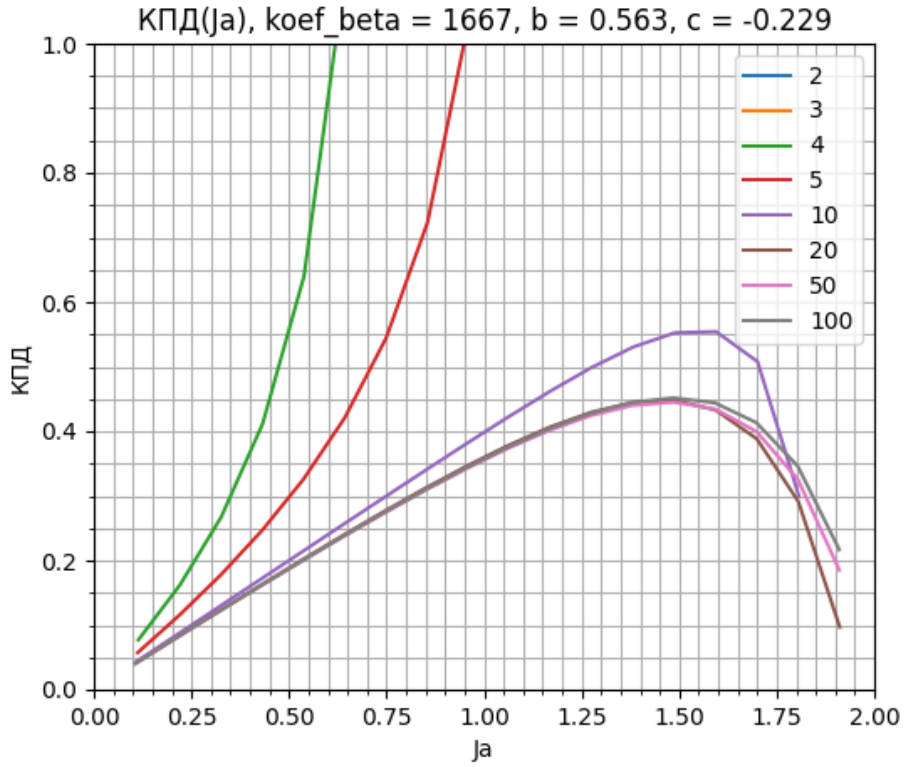


Рисунок 5.3 – График КПД(Ja) при значении коэффициентов для a/L = 100

Результаты для разбегов ниже 10 представляются нефизичными, поэтому было принято решение провести оптимизацию коэффициентов c, b и $k_\beta$ отдельно для каждого режима движения. Результаты такой оптимизации представлены ниже в таблице 5.3.

| a/L | Ja | koef_β | c | b | КПД |
|---|---|---|---|---|---|
| 2 | 0.239 | 2.505 | 0.030 | 0.901 | 1.332 |
| 3 | 0.266 | 2.505 | 0.030 | 0.701 | 1.217 |
| 4 | 0.279 | 3.505 | 0.030 | 0.501 | 1.603 |
| 5 | 0.446 | 6.005 | 0.040 | 0.401 | 1.696 |
| 10 | 1.337 | 9.005 | 0.040 | 0.601 | 1.262 |
| 20 | 1.379 | 296.757 | 0.029 | 0.532 | 0.542 |
| 50 | 0.937 | 684.362 | 0.421 | 0.674 | 0.412 |
| 100 | 1.051 | 478.540 | 0.300 | 0.676 | 0.442 |

Таблица 5.3 – Результаты оптимизации по koef_β, c, b и J для каждого a/L

Как видно из результатов, оптимизация проходит успешно на разбегах a/L = 20, 50, 100. КПД на данных значениях разбегов увеличивается до 44%. Результаты для оставшихся разбегов неудовлетворительны, следовательно их применение невозможно. На рисунке 5.4 представлены результаты оптимизации.

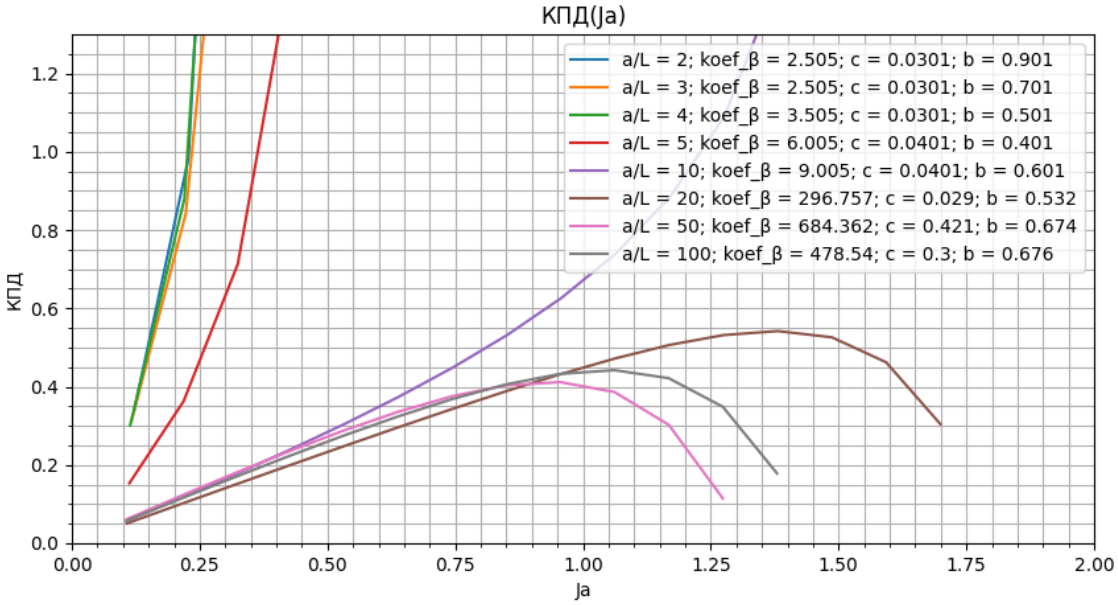


Рисунок 5.4 – График КПД(Ja) при найденных для каждого a/L коэффициентах

Так как при оптимизации по всем параметрам результаты для разбегов 2, 3, 4, 5, 10 не являются физичными, то необходимо подобрать ограничения для коэффициентов c, b и $k_\beta$ для данных разбегов. Далее была проведена оптимизация по c, b, $k_\beta$ и J с учетом подобранных ограничений. Результаты оптимизации представлены в таблице 5.4 и рисунке 5.5.

| a/L | Ja | koef_β | c | b | КПД |
|---|---|---|---|---|---|
| 2 | 0.546 | 1.200 | 0.030 | 0.901 | 0.271 |
| 3 | 0.859 | 1.200 | 0.020 | 0.901 | 0.312 |
| 4 | 1.175 | 1.250 | 0.030 | 0.900 | 0.541 |
| 5 | 1.279 | 1.400 | 0.025 | 0.950 | 0.522 |
| 10 | 1.331 | 6.252 | 0.031 | 0.810 | 0.530 |
| 20 | 1.379 | 296.757 | 0.029 | 0.532 | 0.542 |
| 50 | 0.937 | 684.362 | 0.421 | 0.674 | 0.412 |
| 100 | 1.051 | 478.540 | 0.300 | 0.676 | 0.442 |

Таблица 5.4 – Результаты оптимизации с условиями

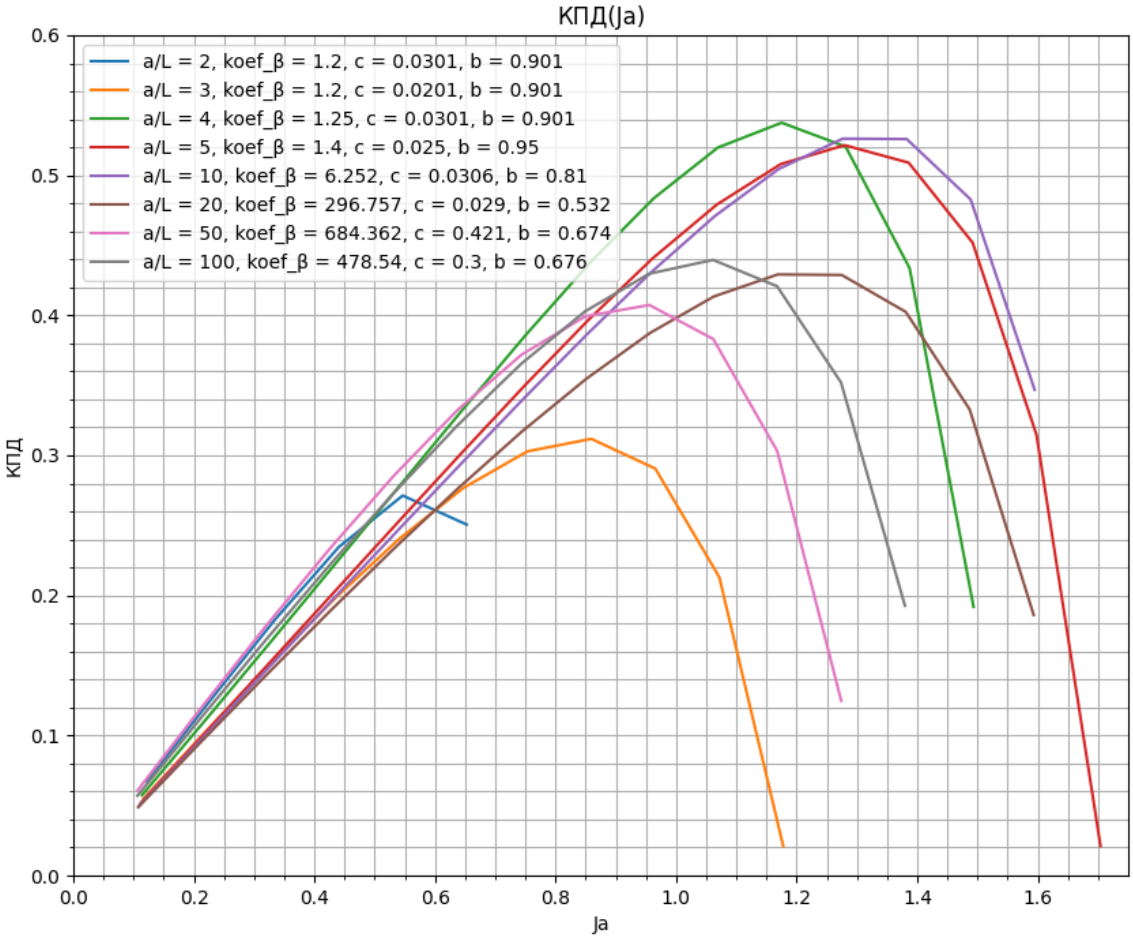


Рисунок 5.5 – Результаты оптимизации с условиями

Далее, в таблице 5.5 и рисунке 5.6, показано сравнение КПД при исходном значении коэффициентов и рассчитанном с помощью оптимизации с ограничениями.

| | Значения КПД и J при исходных значениях коэффициентов | | Значения КПД и J при оптимизации по коэффициентам с условиями | | % увеличения КПД |
|---|---|---|---|---|---|
| a/L | Ja | КПД | Ja | КПД | % |
| 2 | 0.499 | 0.120 | 0.546 | 0.271 | 15.1 |
| 3 | 0.675 | 0.154 | 0.859 | 0.312 | 15.8 |
| 4 | 0.835 | 0.206 | 1.175 | 0.541 | 33.5 |
| 5 | 0.956 | 0.252 | 1.279 | 0.522 | 27 |
| 10 | 1.160 | 0.325 | 1.331 | 0.530 | 20.5 |
| 20 | 1.205 | 0.331 | 1.379 | 0.542 | 21.1 |
| 50 | 1.241 | 0.342 | 0.937 | 0.412 | 7 |
| 100 | 1.256 | 0.349 | 1.051 | 0.442 | 9.3 |

Таблица 5.5 – Результаты и сравнение КПД

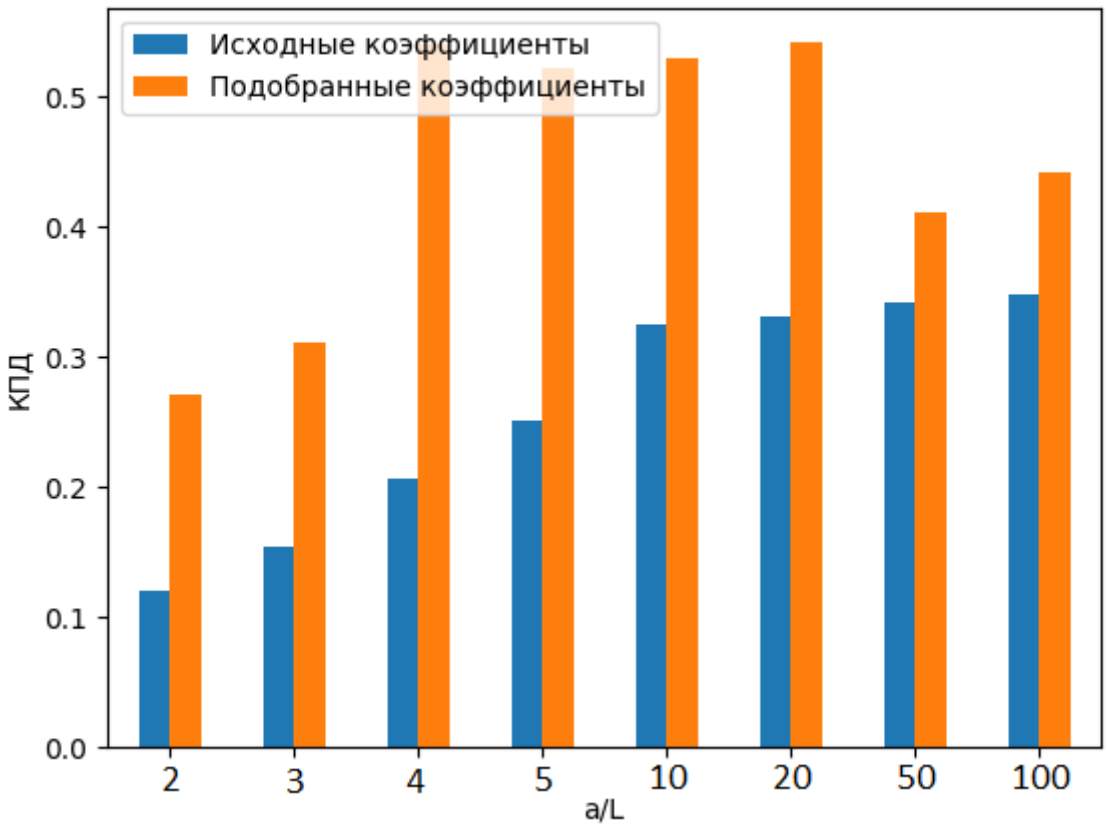


Рисунок 5.6 – Сравнение КПД

# ЗАКЛЮЧЕНИЕ

В дипломной работе было произведена оптимизация функции расчета КПД ГЛД. В качестве метода оптимизации был выбран метод многопараметрической оптимизации нулевого порядка – метод Хука-Дживса. Данный метод был протестирован на функции Розенброка и был получен результат, что данный метод работает корректно.

Оптимизация проводилась по 4 параметрам при 8 значениях разбега a/L: поступи J, $k_\beta$ - коэффициенту скорости поворота лопатки ГЛД, а также коэффициентам «b» и «c», которые участвуют в расчете закона перемещения и отвечают за симметричность закона перемещения относительно T/2 и «полноту» синусоиды соответственно.

Полученные результаты оптимизации полностью не удовлетворяют теоретическим представлениям. Для значений больших значений разбегов оптимизация проводилась безупречно и полученные результаты полностью удовлетворительны. Однако при малых значениях разбега результаты оптимизации являются нефизичными и их применение невозможно.

Так как часть результатов оптимизации была неудовлетворительной, был проведен анализ и установлены ограничения при оптимизации для коэффициентов, при которых КПД увеличивается и результаты, для всех значений разбега, являются пригодными и физичными.

Однако, в любом случае, данный вопрос требует дальнейшего исследования.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. **Артюшков Л.С. Ачкинадзе А.Ш., Русецкий А.А.** Судовые движетели - Л. : Судостроение А, 1988. - стр. 296.
2. **Бхаргава А.** Грокаем алгоритмы. Иллюстрированое пособие для программистов. - СПб. : Питер, 2021. - стр. 29-40.
3. **Войткунский И.Я.** Справочник по теории корабля 1 т.: Гидромеханика. Сопротивление движению судов. Судовые движители. - Л. : Судостроение, 1985. - Т. 1.
4. **Гребенникова И.В.** Методы оптимизации - Екатеринбург : УрФУ, 2017. - стр. 148.
5. **Захарова Е.М. Минашина И.К.** Обзор методов многомерной оптимизации // Информационные процессы - М. : ИППИ РАН, 2014 г.. - 3 : Т. 14. - стр. 256-274.
6. **Короткин А.И.** Присоединенные массы судна. Справочник. - Л. : Судостроение, 1986.
7. **Кочегурова Е.А.** Теория и методы оптимизации - Томск : ТПУ, 2012. - стр. 157.
8. **Лемешко Б.Ю.** Методы оптимизации - Новосибирск : НГТУ, 2009. - стр. 126.
9. **Попова Т.М.** Методы безусловной оптимизации: Тексты лекций - Хабаровск : ТОГУ, 2013. - стр. 76.
10. **Прокопенко Н.Ю.** Методы оптимизации - Нижний Новгород : ННГАСУ, 2018. - стр. 118.
11. **Хук Р. Дживс Т.А.,** Прямой поиск решения для числовых и статических проблем. - 1961. - стр. 212-219.
12. **Чепурко С.И. Яковлев А.Ю.** Численное исследование работы гребных лопаток // Морские интеллектуальные технологии. - СПб., 2021 г.. - 3 : Т. 1. - стр. 34-40.
13. **Шипилов С.А.** Методы безусловной многомерной оптимизации - Новокузнецк : НФИ КемГУ, 2000. - стр. 31.
14. **Яковлев А.Ю. Тхант Зин** Полуэмпирический метод оценки характеристик гидродинамических профилей // Морские интеллектуальные технологии. - СПб., 2019 г.. - 4 : Т. 1. - стр. 29-33.

15. **Яковлев А.Ю. Чепурко С.И.,** Гребной лопастной движетель подводного аппарата, патент на полезную модель : 209324. - 14 10 2021 г.
16. **Carlton J.S.** Marine propellers and propulsion, Elsevier, 2012. - Third edition.
17. **Fuhrer C. Solem J.E., Verdier O.** Scientific Computing with Python 3: An example-rich, comprehensive guide for all of your Python computational needs: Packt Publishing, 2016. - стр. 332.
18. **Python** https://www.python.org.
19. **Rosenbrock H.H.** An Automatic Method for Finding the Greatest or Least Value of a Function // The Computer Journal. - 1960 г.. - 3 : Т. 3. - стр. 175-184.
20. **Van Buren T. Floryan D., Smits A.J.,** Bio-inspired underwater propulsors: arXiv:1801.09714v2 [physics.flu-dyn] 5 Jul 2018.

# ПРИЛОЖЕНИЯ

## Приложение А. Код функции расчета КПД

```
import math


def calculations(x1, a_L1, mode='old', conditions=False) -> float:

    # Пустые массивы
    t_T = []  # массив времени
    x = []  # перемещение
    x_zvezd = []  # первая производная по перемещению
    x_zvezd_2 = []  # вторая производная по перемещению
    beta = []  # угол поворота
    beta_zvezd = []  # первая производная по углу поворота
    beta_zvezd_2 = []  # вторая производная по углу поворота
    X = []  # гидродинамическая составляющая силы
    Xi = []  # инерционная составляющая силы
    P = []  # упор
    dA = []  # работа

    # Исходные данные
    J = x1[0]  # поступь
    k = 2 / 0.93  # коэффициент
    v0 = 1  # отнормированная скорость набегающего потока
    e_L = 0.1  # относительная толщина профиля
    a_L = a_L1  # относительный разбег
    koefficent = x1[1]
    f_L = 0  # кривизна
    dt_T = 0.01  # относительный шаг по времени
    Re = 1e6  # Число Рейнольдса
    Sh = math.pi / J  # число Струхаля
    mu = (0.77 * e_L) / (1 - 0.6 * e_L)  # функция относительной тол-
                                           щины профиля
    A_L = 1 / (2 * (1 + mu ** 2))
    alpha = math.atan(2 * f_L * (1 + mu ** 2))
    R = (1 + mu) / math.cos(alpha)
    r = R / (1 + 2 * mu)

    # Расчет присоединенных масс
    mu_11 = (math.pi / 4) * (A_L ** 2) * (r ** 2 + R ** 2 - 2 *
            math.cos(2 * alpha))
```

```
mu_22 = (math.pi / 4) * (A_L ** 2) * (r ** 2 + R ** 2 + 2 *
        math.cos(2 * alpha))
mu_12 = (math.pi / 4) * (A_L ** 2) * math.sin(alpha)
mu_16 = (math.pi / 8) * (A_L ** 3) * (r ** 2 + R ** 2 + 4 * (r +
        R) * math.cos(alpha)) * math.sin(alpha)
mu_26 = (math.pi / 8) * (A_L ** 3) * (r ** 3 + R ** 3 + (r ** 2 +
        R ** 2) * math.cos(alpha) + 2 * (r + R) * math.cos(2 *
        alpha))

# Заполнение массива по времени
for i in range(0, 105):
    t_T.append(float("{0:.2f}".format(i * dt_T)))

# Заполнение массива по перемещению
n = 1.5
for t in t_T:
    if mode == 'old':
        c = 0
        b = 1
    elif mode == 'new':
        c = x1[2]
        b = x1[3]
    x.append((a_L/2)*(1-((c+math.cos(2*math.pi*t+((2*c)/n)+n))*
             math.cos(n)+(b*math.sin(2*math.pi*t+((2*c)/n)+n))*
             math.sin(n))/(math.sqrt((c+math.cos(2*math.pi*t+((2*c/
             n)+n))**2+(b*math.sin(2*math.pi*t+((2*c)/n)+n))**2))))

# Заполнение массива первых производных по перемещению
for j in range(len(x) - 1):
    if j == 0:
        x_zvezd.append(0.0)
    elif j != 0:
        x_zvezd.append((x[j + 1] - x[j - 1]) / (2 * dt_T))

# Заполнение массива вторых производных по перемещению
for j in range(len(x_zvezd) - 1):
    if j == 0:
        x_zvezd_2.append(0.0)
    elif j != 0:
        x_zvezd_2.append((x_zvezd[j + 1]-x_zvezd[j-1])/(2 * dt_T))

# Заполнение массива углов поворота
for i in range(len(t_T) - 1):
    beta.append(0.5*((math.pi/2)+math.atan(v0-koefficent*
                x_zvezd[i])))
```

```
# Заполнение массива первых производных по углу поворота
for j in range(len(beta)-1):
    if j == 0:
        beta_zvezd.append(0.0)
    elif j != 0:
        beta_zvezd.append((beta[j+1]-beta[j-1])/(2*dt_T))

# Заполнение массива вторых производных по углу поворота
for j in range(len(beta_zvezd) - 1):
    if j == 0:
        beta_zvezd_2.append(0.0)
    elif j != 0:
        beta_zvezd_2.append((beta_zvezd[j+1]-beta_zvezd[j-1])/(2
                            *dt_T))

# Расчет гидродинамической составляющей силы
for i in range(len(x_zvezd)):
    # Коэффициент нормальной силы на пластине (Формула Релея)
    C_n = 2*math.pi*((math.sin(abs(beta[i])))/(4+math.pi*
          (math.sin(abs(beta[i])))) * (1-0.5+(1.6/(abs(beta[i])*
          (6/math.pi))**0.7))

    X.append(k*(-(1+Sh*x_zvezd[i])*abs(1+Sh*x_zvezd[i])*(C_n *
             math.sin(beta[i])+(((2+2.4*e_L+17*e_L**3)*0.455)/
             ((math.log10(Re*abs(1- x_zvezd[i])))**2.58)))))

# Расчет инерционной составляющей силы
for i in range(len(beta_zvezd_2)):
    Xi.append(-1*(-2*Sh**2*(x_zvezd_2[i]*(mu_12-mu_11*
              (math.cos(beta[i]))**2-mu_22*(math.sin(beta[i]))**2)+
              x_zvezd[i]*beta_zvezd[i]*(math.sin(2*beta[i])*(mu_11-
              mu_22)+2*math.cos(2*beta[i])*mu_12)-beta_zvezd_2[i]*
              (math.cos(beta[i])*mu_16-math.sin(beta[i])*mu_26)+
              beta_zvezd[i]**2*(math.sin(beta[i])*mu_16+
              math.cos(beta[i]) * mu_26))))
Xi[0] = Xi[-2]
Xi[1] = Xi[-1]

# Расчет упора
for i in range(len(Xi)):
    P.append(Xi[i] + X[i])

# Расчет КПД
for i in range(len(P)):
    dA.append(x_zvezd[i] * dt_T * P[i])  # работа
```

```
A = -sum(dA)  # суммарная работа
P_average = sum(P) * dt_T  # средний по периоду упор
kpd = P_average / (Sh * A)  # КПД


# условия для оптимизации
if conditions == True:
    if b > 1 or b <= -1: kpd = 1e-37
    if c >= 1 or c <= -1: kpd = 1e-38
    if kpd <= 0: kpd = 1e-39
    if kpd >= 1: kpd = 1e-40


return kpd
```

## Приложение Б. Код метода оптимизации

```
machineAcc = 1e-9

# Исследующий поиск
def utilSearch(b, h, f, a_L, mode, conditions):
    bres = b[:]
    fb = f(bres, a_L, mode, conditions)
    for i in range(0, len(bres)):
        bn = bres
        bn[i] = bn[i] + h[i]
        fc = f(bn, a_L, mode, conditions)
        if (fc + machineAcc >= fb):
            bres = bn
            fb = fc
        else:
            bn[i] = bn[i] - 2 * h[i]
            fc = f(bn, a_L, mode, conditions)
            if (fc + machineAcc >= fb):
                bres = bn
                fb = fc
    return bres


Path1 = []
Path2 = []
Path3 = []
Path4 = []


# Метод конфигураций Хука-Дживса
def HJ(b1, h, e, f, a_L, mode, conditions):
    z = 0.01
    runOuterLoop = True
    while (runOuterLoop):
        runOuterLoop = False
        runInnerLoop = True
        xk = b1  # step1
        b2 = utilSearch(b1, h, f, a_L, mode, conditions)  # step2
        Path1.append(b1)
        Path2.append(b2)
        Path3.append(xk)
        while (runInnerLoop):
            Path1.append(b1)
            Path2.append(b2)
```

```
        runInnerLoop = False
        for i in range(len(b1)):  # step3
            xk[i] = b1[i] + 2 * (b2[i] - b1[i])
        Path3.append(xk)
        x = utilSearch(xk, h, f, a_L, mode, conditions)  # step4
        Path4.append(x)
        b1 = b2  # step5

        fx = f(x, a_L, mode, conditions)
        fb1 = f(b1, a_L, mode, conditions)
        if (fx + machineAcc < fb1):  # step6
            b2 = x
            runInnerLoop = True  # to step3
        elif (fx - machineAcc > fb1):  # step7
            runOuterLoop = True  # to step1
            break
        else:
            s = 0
            for i in range(len(h)):
                s += h[i] * h[i]
            if (e * e + machineAcc > s):  # step8
                break  # to step10
            else:
                for i in range(len(h)):  # step9
                    h[i] = h[i] * z
                runOuterLoop = True  # to step1
return b1  # step10
```