**Министерство образования и науки Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования «КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ» ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ**

Специальность: «01.03.02 Прикладная математика и информатика»

**Курсовая работа по курсу «Численные методы»**

**на тему:**

**«Задача о стабилизации экраноплана»**

Выполнил:

Студент группы 09-812

Садыков И. Н.

Проверил:

Макаров М.В.

КАЗАНЬ – 2021 г.

Содержание

[Введение 3](#_Toc90830561)

[Постановка задачи 4](#_Toc90830562)

[Описание метода 7](#_Toc90830563)

[Тестирование программы для интегрирования задачи Коши 9](#_Toc90830564)

[Задача стабилизации экраноплана 18](#_Toc90830565)

[Вывод 35](#_Toc90830566)

[Список использованной литературы 36](#_Toc90830567)

[Исходный код 37](#_Toc90830568)

# Введение

Экраноплан - летательный аппарат, использующий эффект существенного увеличения подъемной силы при движении вблизи экрана (поверхности воды, земли, снега или льда). При равных массе и скорости удлинение [крыла](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D1%8B%D0%BB%D0%BE_%D1%81%D0%B0%D0%BC%D0%BE%D0%BB%D1%91%D1%82%D0%B0) экраноплана намного меньше, чем у [самолёта](https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%BC%D0%BE%D0%BB%D1%91%D1%82). По международной классификации ([ИМО](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D0%B4%D1%83%D0%BD%D0%B0%D1%80%D0%BE%D0%B4%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D1%80%D1%81%D0%BA%D0%B0%D1%8F_%D0%BE%D1%80%D0%B3%D0%B0%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F)) экранопланы относят к [морским судам](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D1%80%D1%81%D0%BA%D0%BE%D0%B9_%D1%82%D1%80%D0%B0%D0%BD%D1%81%D0%BF%D0%BE%D1%80%D1%82). Одной из основных проблем, которая препятствует серийному производству экранопланов, является их продольная неустойчивость, вызванная необратимым процессом увеличения угла кабрирования, что может привести к перевороту судна. Кабрирование – это резкое увеличение угла тангажа с неуправляемым набором высоты, что часто заканчивается аварией или катастрофой. Проблема обеспечения устойчивости экранопланов может быть решена аэродинамической стабилизацией – обеспечением самостабилизации за счёт выбора специальной геометрии несущих поверхностей и их компоновки в аэродинамической схеме.

Цель данной работы исследовать оптимальный набор параметров и начальных условий необходимых для конструирования экраноплана с высоким показателем стабилицизации.

# 

# Постановка задачи

**Задача о стабилизации экраноплана**

Экраноплан - летательный аппарат, использующий эффект существенного увеличения подъемной силы при движении вблизи экрана (поверхности воды, земли, снега или льда). При этом возникает опасность возникновения сильных колебаний по высоте полета. Таким образом, необходимо выбрать конструктивные параметры, обеспечивающие устойчивость аппарата. Уравнения движения экраноплана в вертикальной плоскости при определенных допущениях могут быть записаны в виде [1]:

Здесь - ускорение силы тяжести, - модуль скорости полета, θ - угол наклона траектории, - угол между продольной осью экраноплана и экраном (угол тангажа), H - отклонение высоты полета от заданной, - продольная перегрузка, - вертикальная перегрузка, - продольный момент внешних сил, - продольный момент инерции аппарата.

Для , , принимают следующие зависимости:

где - плотность воздуха, - площадь крыла, - вес аппарата, - коэффициент лобового сопротивления, - угол атаки, - коэффициенты, характеризующие взаимодействие аппарата с потоком воздуха, - плечо горизонтального оперения.

Введением новой неизвестной система (1) сводится к системе из пяти обыкновенных дифференциальных уравнений первого порядка.

**Задание**

1. Напишите программу интегрирования задачи Коши для системы из n уравнений первого порядка вида

на произвольном отрезке используя метод Рунге – Кутты 3-го порядка точности с постоянным шагом h:

2. Протестируйте программу на примере системы уравнений

на отрезке с точным решением (проверьте!)

3. Для тестовой задачи постройте графики зависимости максимальной погрешности решения и от шага h. Что следует из полученных графиков?

4. Решая систему уравнений движения экраноплана при помощи разработанной программы, рассчитайте зависимость высоты полета от времени на интервале . Результаты расчетов оформить в виде графиков . Параметры аппарата и другие исходные данные (система единиц измерения — СИ):

Начальные условия:

Обратите внимание на то, что H(t) быстро меняется в начальные моменты времени. Поэтому в окрестности начала отрезка интегрирования необходимо выбирать существенно меньший шаг интегрирования h, чем в другие моменты времени.

5. Исследуйте влияние конструктивных параметров и начальных условий полета на характер устойчивости аппарата, изменяя - плечо горизонтального оперения - в пределах [2, 10], начальную скорость в интервале [50, 100].

# Описание метода

Рассмотрим задачу Коши для системы обыкновенных дифференциальных уравнений первого порядка.

на равномерной с шагом h сетке узлов:

Используя формулу Тейлора с остаточным членом в форме

Лагранжа, для каждого получим разложение

Используем равенство , а так же то, что вторую производную можно аппроксимировать разделенной разностью 2-го порядка, то есть: и 3-ю производную можно аппроксимировать разделенной разностью 3-го порядка.

Тогда метод Рунге-Кутты 3-го порядка описывается следующим образом:

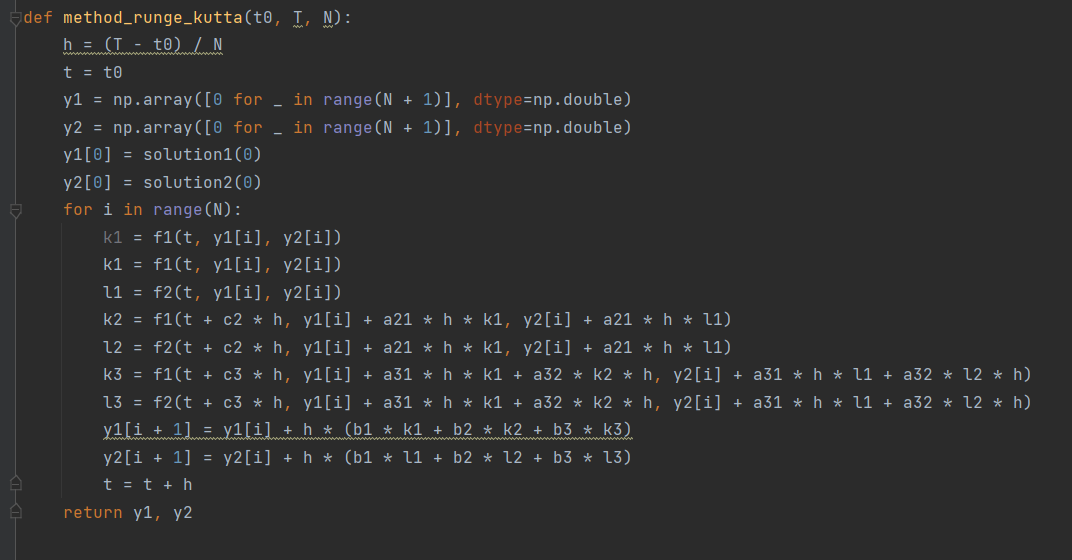
*,*

где

Для определения восьми коэффициентов метода при 3-м порядке метода Рунге-Кутты используется система из следующих 6 уравнений:

# Тестирование программы для интегрирования задачи Коши

Задача 1. Интегрирование задачи Коши для системы из n уравнений первого порядка.



Функция method\_runge\_kutta принимает на вход три параметра: границы интервала, на котором надо найти решение ОДУ, и количество разбиений.

Задача 2.

Дана система уравнений:

Её точное решение

Проверим, что действительно являются точными решениями:

Таким образом указанные решение являются точными.

Задача 3. Тестирование метода Рунге-Кутты для заданной системы уравнений.

на отрезке , точное решение которой:

в программной реализации метод для данной системы будет выглядеть следующим образом:

Тестирование программы осуществляется на отрезке . Алгоритм тестирования следующий: отрезок разбивается на равномерную сетку узлов, и в узлах сетки программа находит решения методом Рунге-Кутты и точное решение.

Пусть . Величина представляет собой абсолютную

погрешность решения в i-той точке сетки. Обозначим через максимальную погрешность решение,

Приведем графики зависимости максимальной погрешности и от выбранного шага h:

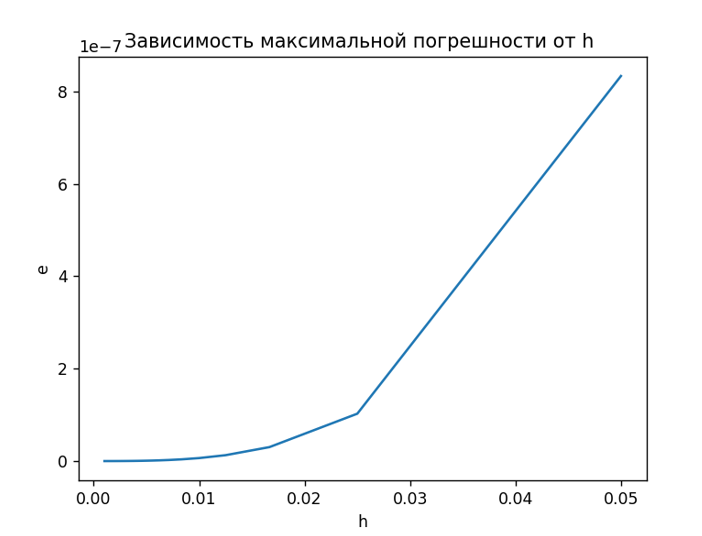


Рис1. График зависимости погрешности от шага h для

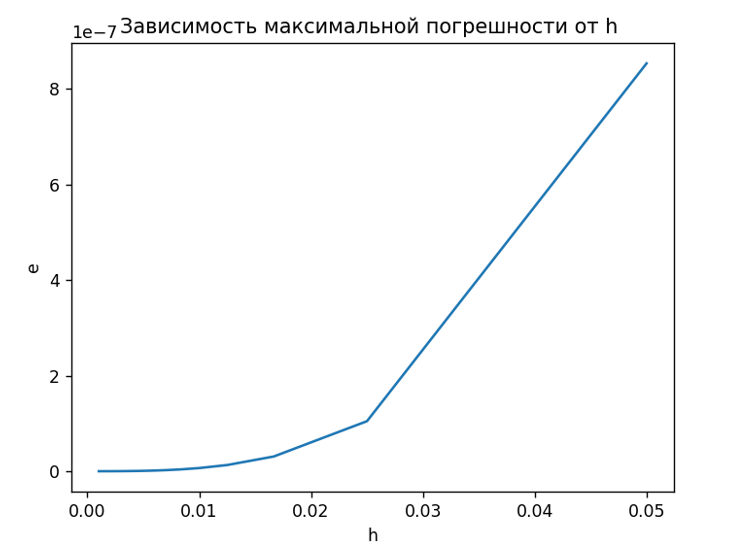


Рис2. График зависимости погрешности от шага h для

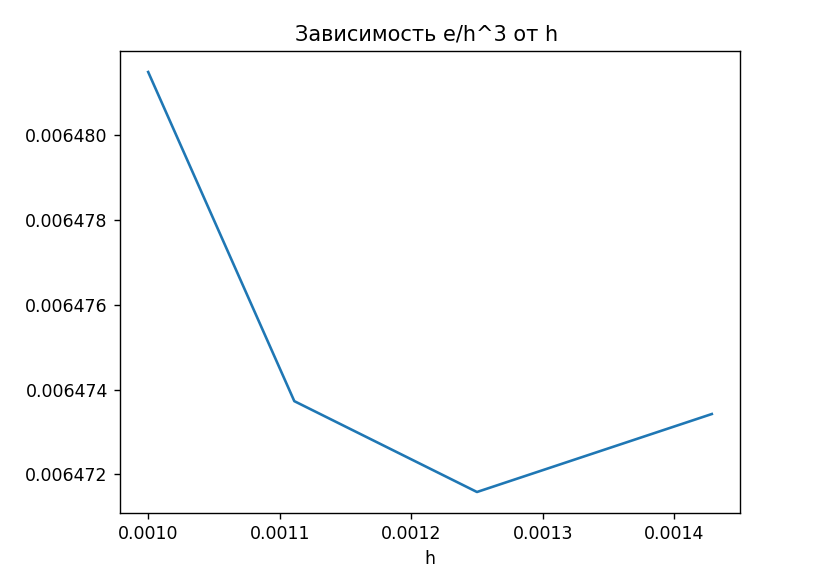


Рис3. График зависимости погрешности от шага h для

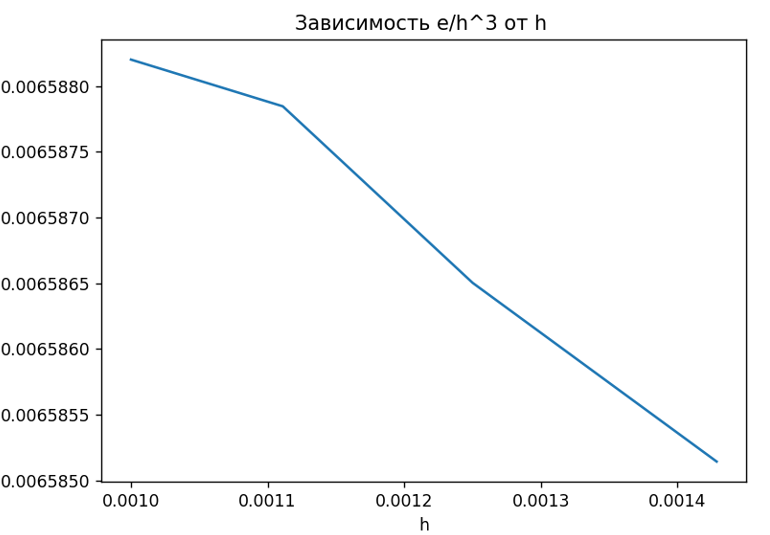


Рис4. График зависимости погрешности от шага h для

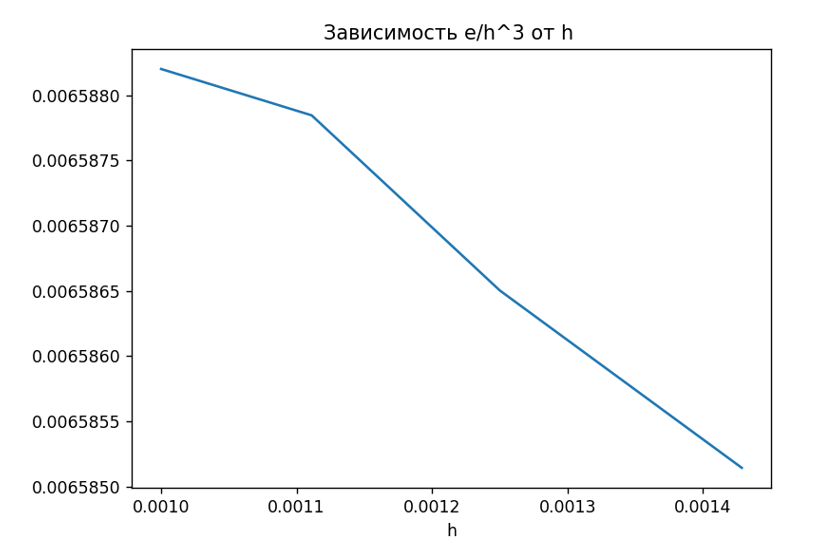


Рис5. График зависимости погрешности от шага h для

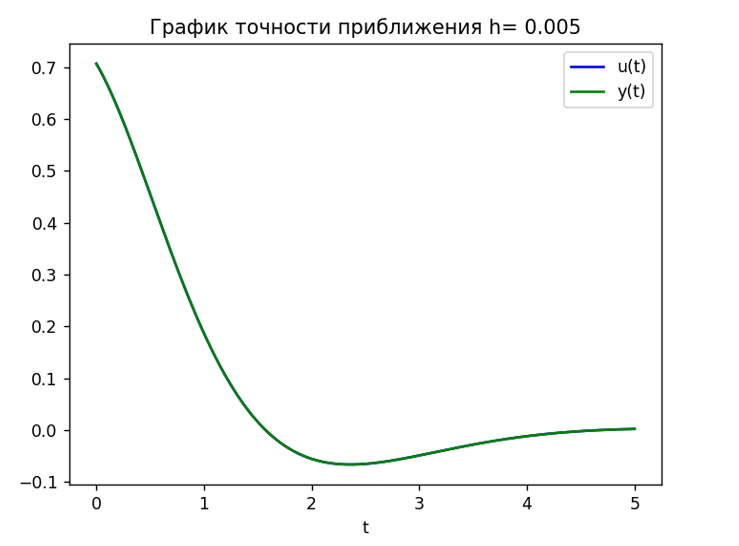


Рис6. График решений, вычисленных методом Рунге-Кутты и точным для

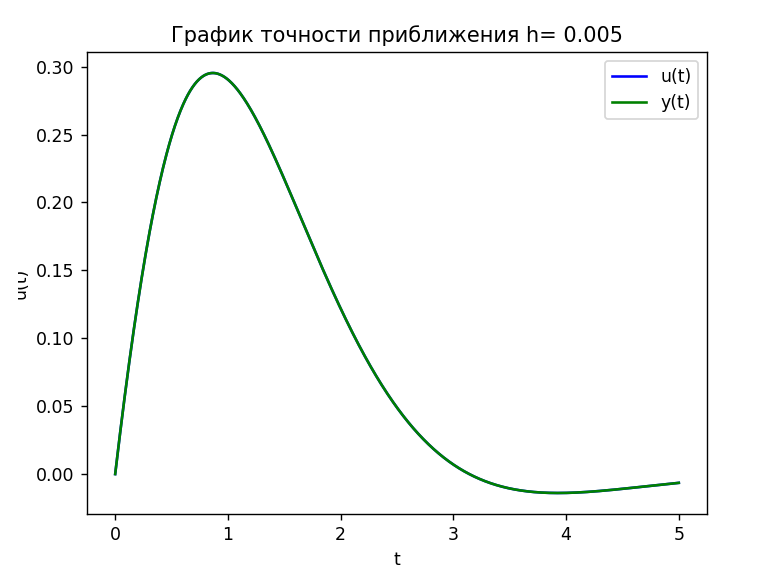


Рис7. График решений, вычисленных методом Рунге-Кутты и точным для

Таким образом, на тестовой задаче убедились, что метод Рунге-Кутты вычисляет решение системы ОДУ верно, и с уменьшением h пропорционально уменьшается максимальная погрешность решения, а также, что погрешность может быть сделана сколь угодно малой при соответствующем выборе h. Параметр, характеризующий устойчивость метода не изменяется с изменением h, что говорит о сходимости метода. Программа применима для решения систем ОДУ.

# Задача стабилизации экраноплана

Задача 4. Решение системы уравнений движения экраноплана при помощи разработанной программы.

Заданная система уравнений

не является системой однородных дифференциальных уравнений, так как есть уравнение второго порядка. Приведем систему к системе ОДУ при помощи новой переменной , тогда и получим следующую систему:

Задаем следующие начальные условия и значения констант в программе:

Начальные условия:

, то есть .

Для наглядности исследуемого процесса стабилизации экраноплана приведем иллюстрацию сил, действующих на экраноплан.

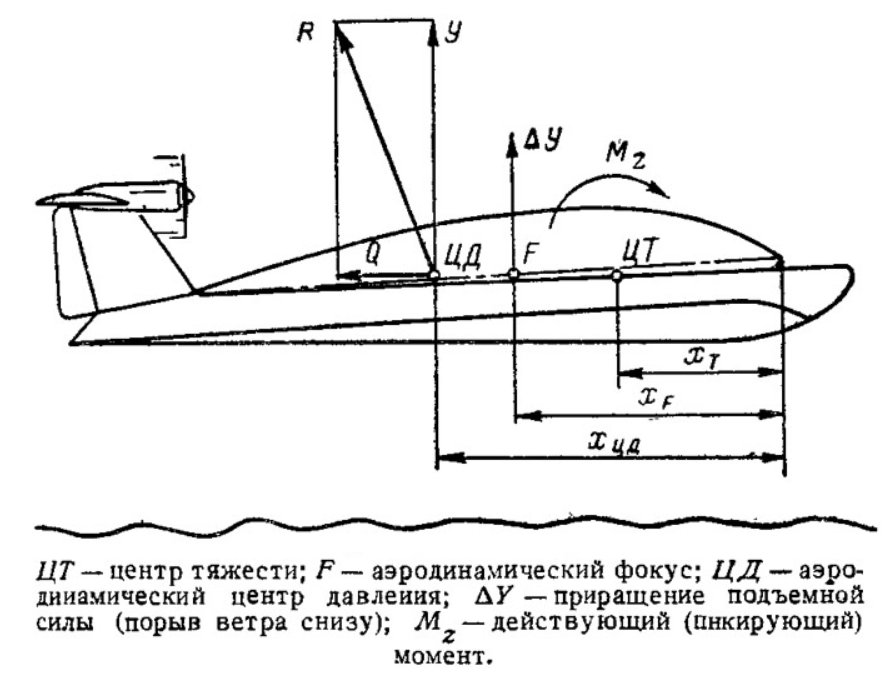


Рис8. Расстановка сил во время полета экраноплана.

При движении крыла с положительным углом атаки поток над ним сильно искривляется его верхней частью и поджимается, что повышает скорость обтекания и, как следствие, над крылом возникает зона пониженного давления. Под крылом, наоборот, происходит торможение потока, уменьшение его скорости, а следовательно и увеличение давления. Таким образом возникает подъемная сила крыла Y. Сумма проекций нормальных и касательных к поверхности профиля крыла сил на ось х дает силу лобового сопротивления Q. Результирующую подъемной силы и силы сопротивления называют полной аэродинамической силой крыла R. Так как R разложено на составляющие по поточным осям, то на рисунке направления этих сил не зависят от угла атаки.

Таким образом можно кратко описать указанные на рисунке силы следующим образом:

- сила тяжести

- подъемная сила крыла.

- сила лобового сопротивления

- полная аэродинамическая сила крыла

*-*порыв ветра снизу

В постановке задачи указано, что есть некоторая заданная высота полета экраноплана и начальная скорость >0. Задача – исследовать стабилизацию экраноплана относительно некоторой заданной высоты с указанными выше начальными условиями. В дальнейшем будем считать, что ось абцисс начала системы координат перенесена на заданную высоту, и отклонение от оси есть отклонение от заданной высоты. Угол тангажа — угол между продольной осью экраноплана и горизонтом.

Исследуем полет экраноплана и поведение искомых функций системы ОДУ в зависимости от времени на интервале .

Рассмотрим изменение модуля скорости.

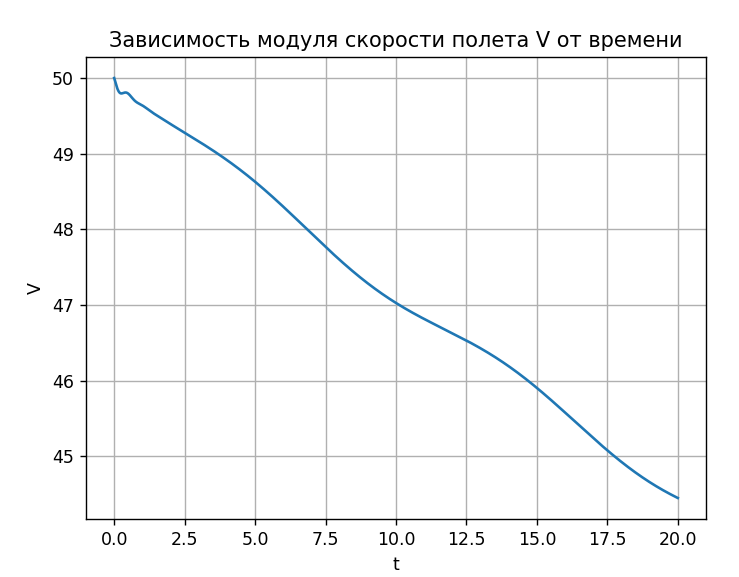


Рис 9. Зависимость модуля скорости от времени при заданных условиях.

Скорость экраноплана со временем уменьшается, так как начальная скорость 50, и аппарату нужно время для набора высоты и применения экранного эффекта.

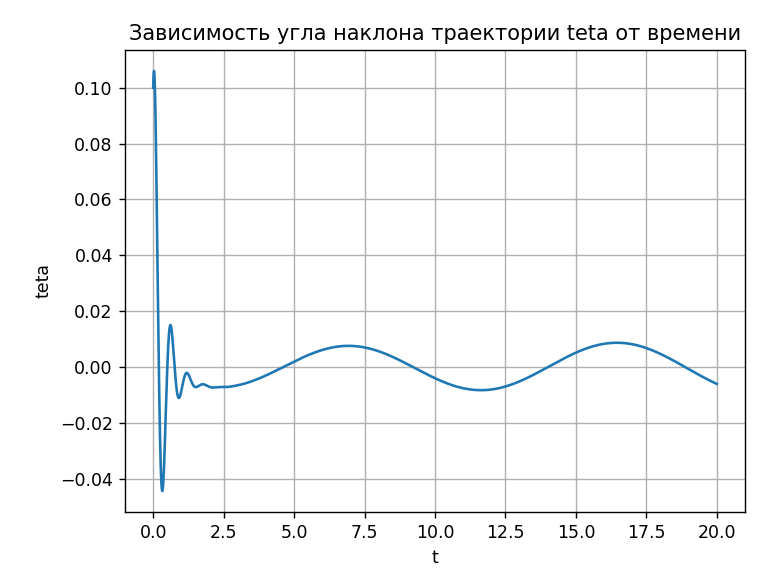


Рис 10. Зависимость угла наклона траектории при заданных условиях.

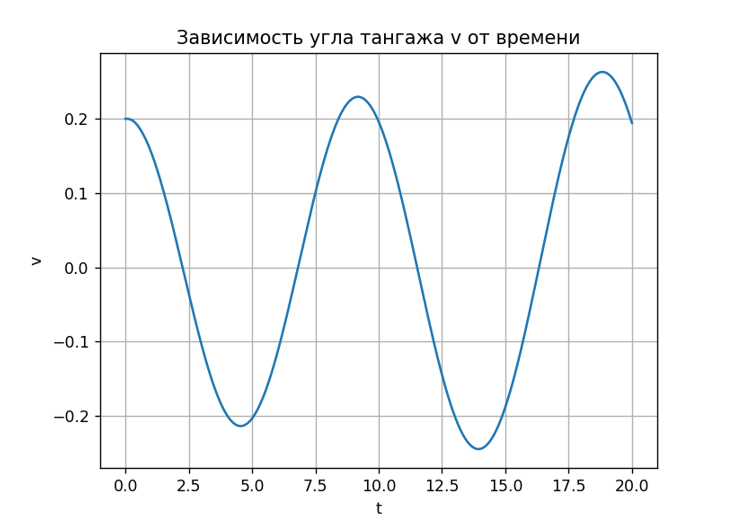


Рис 11. Зависимость угла тангажа от времени при заданных условиях.

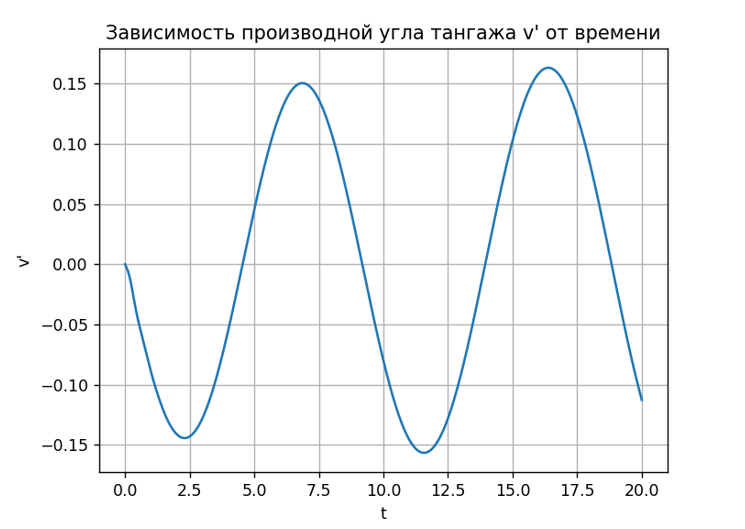


Рис 11. Зависимость производной угла тангажа от времени при заданных условиях.

Изменение значение угла тангажа невелико, экраноплан летит без резких изменений высоты и угла.

Для расчета зависимости высоты полета от времени построим график функции , найденной из системы ОДУ, указанной в задаче.

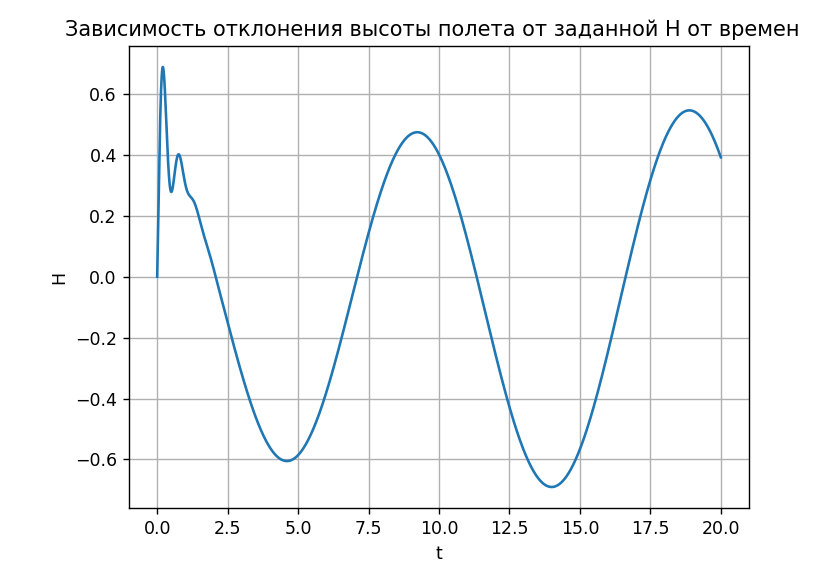


Рис 12. Зависимость отклонения высоты полета от заданной от времени при заданных условиях.

Высота колеблется в небольшом интервале и не сильно отклоняется от заданной высоты, то есть имеет место стабильность полета. Можно заметить, что уменьшение угла тангажа влечет за собой уменьшение высоты, что сходится с определением угла тангажа и его влиянием на высоту.

Задача 5.

Как указывалось ранее задача стабилизации экраноплана можно описать как подбор конструктивных параметром и начальных условий, обеспечивающий наилучшую устойчивость во время полета и плавность полета.

Исследуем влияние такого параметра как — плечо горизонтального оперения — в пределах , начальную скорость в интервале

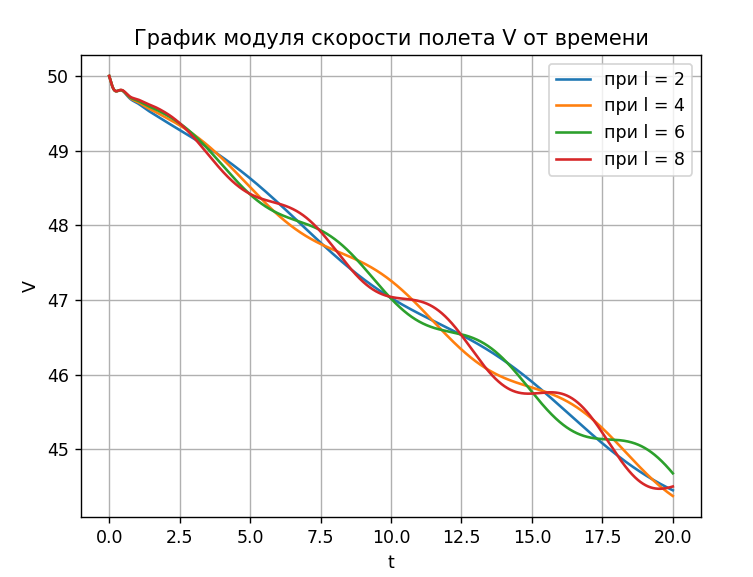


Рис 11. График зависимости модуля скорости от времени при разных .

Можно увидеть, что скорость при , скорость наиболее плавно уменьшается. Уменьшение скорости происходит при всех рассмотренных значениях плеча горизонтального оперения.

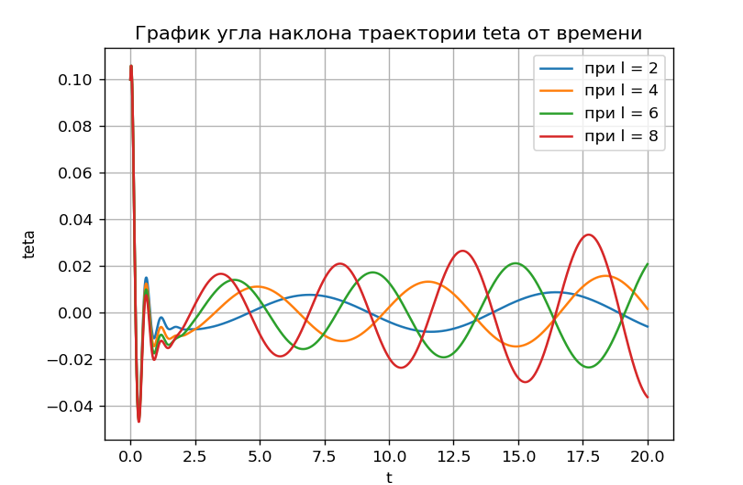
**

Рис 12. График зависимости угла наклона траектории от времени при разных .

Ситуации стабилизации соответствует изменение угла наклона траектории в меньшем интервале, то есть при . Такое значение параметра обеспечивает наибольшую устойчивость полета экраноплана.

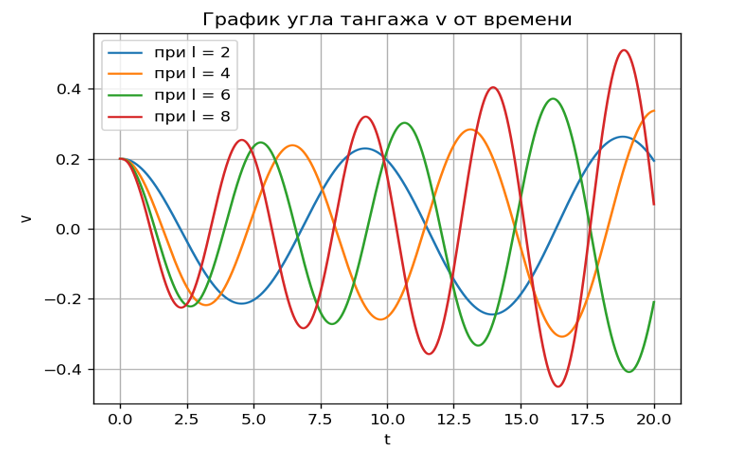


Рис13. График зависимости угла тангажа от времени при разных .

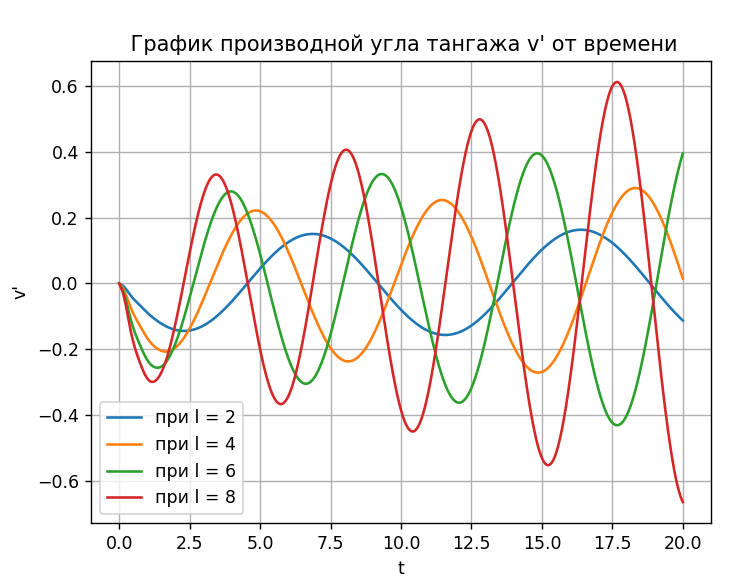


Рис14. График зависимости производной угла тангажа от времени при разных

Интервал изменения угла тангажа меньше всего при . При других значениях разница между максимальным и минимальным значением>1. Период изменения наибольший при .

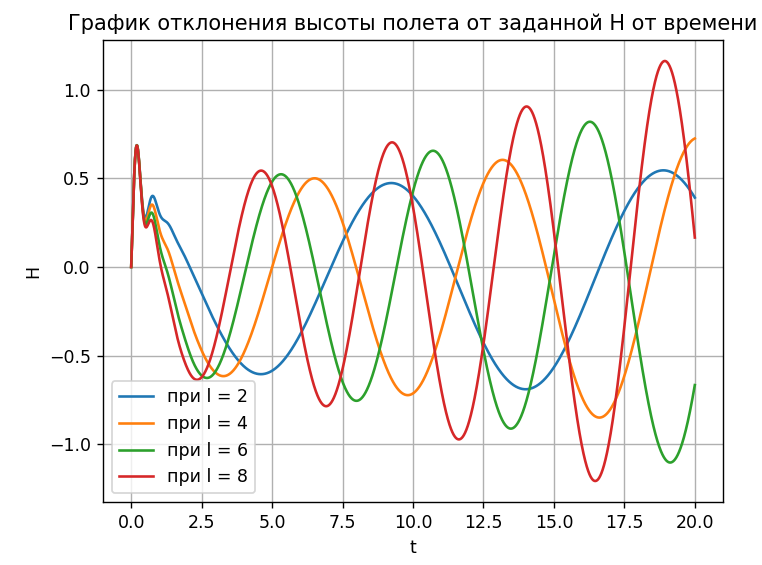


Рис15. График зависимости отклонения высоты полета от времени при разных

При максимальное отклонение высоты полета от заданной меньше всего, и достигается наиболее стабильное положение для полета. При изменении устойчивость экраноплана ухудшается, что можно видеть по графикам исследуемых функций.

Теперь исследуем те же зависимости и значения функций, изменяя в интервале с шагом 10.

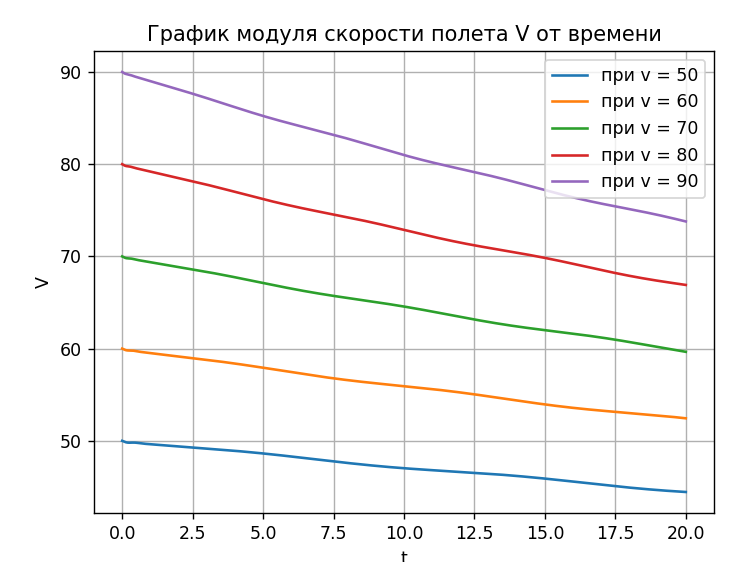


Рис16. График зависимости модуля скорости от времени при разных

Наиболее стабильный модуль скорости во время полета достигается при

и .

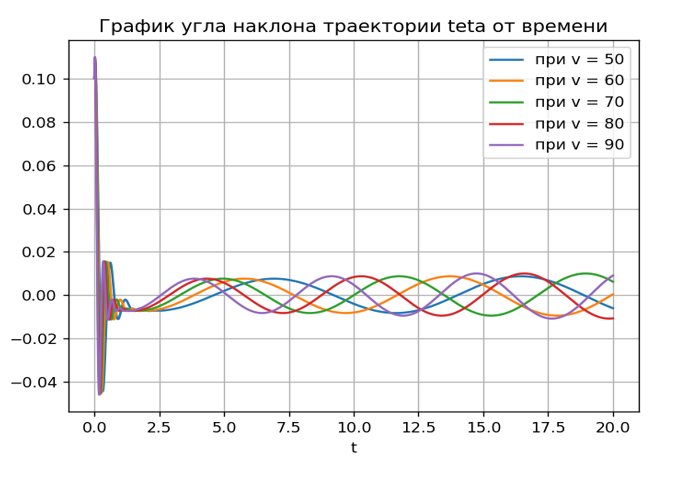


Рис 17. График зависимости угла наклона траектории от времени при разных .

Здесь можно сделать вывод, что для меньшего изменения угла наклона траектории и наибольшей частоты подходит значение .

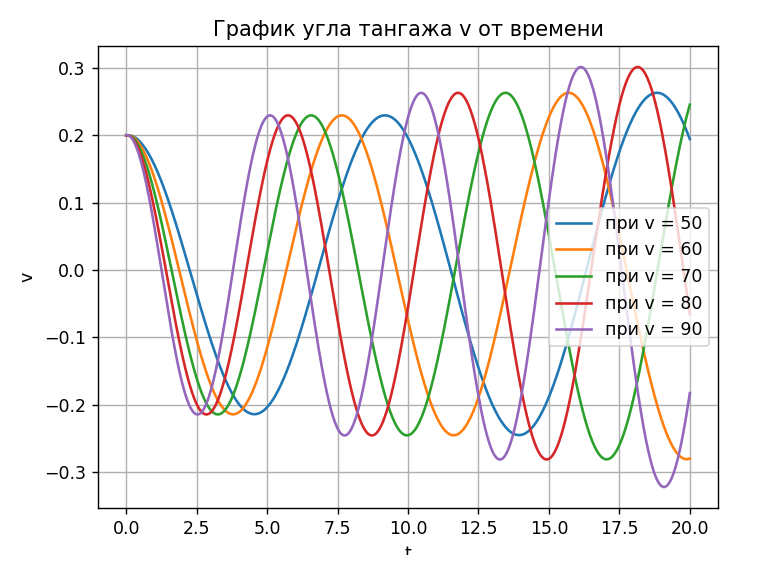


Рис 18. График зависимости угла тангажа от времени при разных .

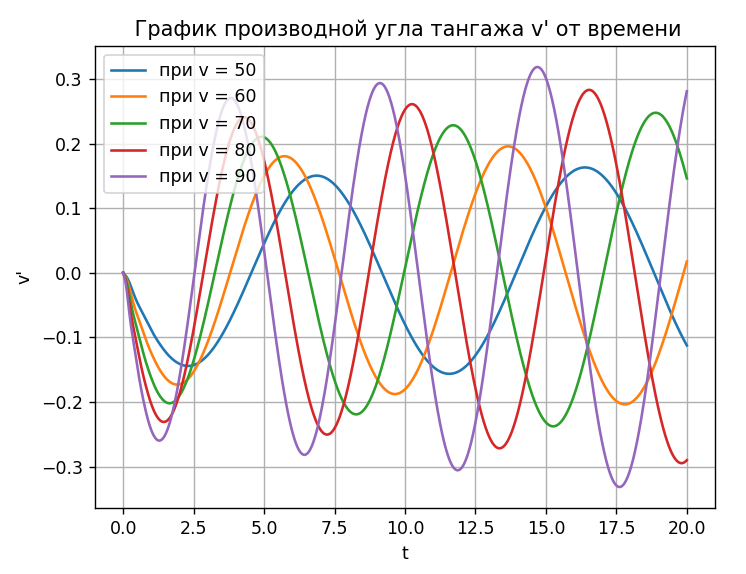


Рис 19. График зависимости производной угла тангажа от времени при разных .

Для стабилизации экраноплана требуется как можно меньшее изменение угла тангажа, поэтому подходят значение  и . Наибольший период достигается при

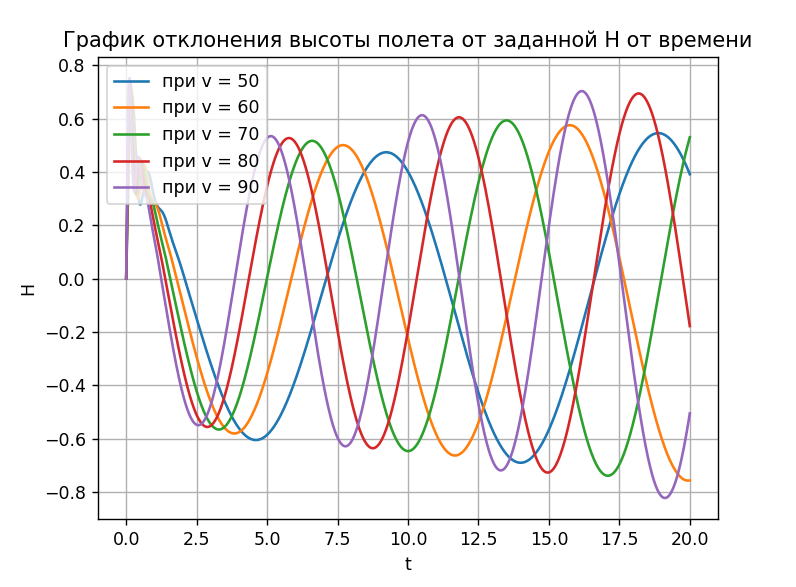


Рис20. График зависимости отклонения высоты полета от времени при разных

Из графиков можно сделать вывод, что стабилизация экраноплана и устойчивость при полете достигается при выборе , так как исследуемые функции меняются в меньшем интервале по сравнению с другими значениями начальной скорости.

# Вывод

Научился решать системы однородных дифференциальных уравнений методом Рунге-Кутты 3-го порядка. Реализована программа для решения систем ОДУ. Проведены исследования задачи стабилизации экраноплана. Получены оптимальные значения для стабилизации экраноплана. Выяснил что относительно заданной высоты стабилизация происходит не сразу, лучшая стабилизация достигается при плече горизонтального оперения длиной 2 и начальной скорости 50. Изменение этих параметров ведет к ухудшению устойчивости аппарата.

# Список использованной литературы

1. Иродов Р.Д. Критерии продольной устойчивости экраноплана // Ученые записки ЦАГИ, т. 1, N 4. С. 63-72.
2. Самарский А.А, Гулин А.В. Численные методы. — М.: Наука, 1989.
3. Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. Численные методы. — М.: Наука,1987
4. Экраноплан / Белавин Н. И. // Чаган — Экс-ле-Бен. — М. : Советская энциклопедия, 1978. — ([Большая советская энциклопедия](https://ru.wikipedia.org/wiki/%D0%91%D0%BE%D0%BB%D1%8C%D1%88%D0%B0%D1%8F_%D1%81%D0%BE%D0%B2%D0%B5%D1%82%D1%81%D0%BA%D0%B0%D1%8F_%D1%8D%D0%BD%D1%86%D0%B8%D0%BA%D0%BB%D0%BE%D0%BF%D0%B5%D0%B4%D0%B8%D1%8F#%D0%A2%D1%80%D0%B5%D1%82%D1%8C%D0%B5_%D0%B8%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5) : [в 30 т.] / гл. ред. [А. М. Прохоров](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%85%D0%BE%D1%80%D0%BE%D0%B2,_%D0%90%D0%BB%D0%B5%D0%BA%D1%81%D0%B0%D0%BD%D0%B4%D1%80_%D0%9C%D0%B8%D1%85%D0%B0%D0%B9%D0%BB%D0%BE%D0%B2%D0%B8%D1%87) ; 1969—1978, т. 29).
5. Белавин, Н. И. Экранопланы : (по данным зарубежной печати). — 2-е изд. — Л. : Судостроение, 1977. — 232 с. — 4500 экз.

## Исходный код

test.py

import numpy as np

import matplotlib.pyplot as plt

c2 = 1 / 3

a21 = 1 / 3

c3 = 2 / 3

b2 = 0

b1 = 1 / 4

b3 = 3 / 4

a31 = 0

a32 = 2 / 3

def f1(t, y1, y2):

return -np.sin(t) / (np.sqrt(1 + np.exp(2 \* t))) + y1 \* (y1 \* y1 + y2 \* y2 - 1)

def f2(t, y1, y2):

return np.cos(t) / (np.sqrt(1 + np.exp(2 \* t))) + y2 \* (y1 \* y1 + y2 \* y2 - 1)

def solution1(t):

return np.cos(t) / (np.sqrt(1 + np.exp(2 \* t)))

def solution2(t):

return np.sin(t) / (np.sqrt(1 + np.exp(2 \* t)))

def initial\_solution\_y(t0, T, N):

h = (T - t0) / N

u = []

u1 = []

for i in range(N + 1):

u.append(solution1(t0 + h \* i))

u1.append(solution2(t0 + h \* i))

return np.array(u), np.array(u1)

def count\_error(y, u\_real):

return max([abs(u\_real[i] - y[i]) for i in range(len(u\_real))])

def method\_runge\_kutta(t0, T, N):

h = (T - t0) / N

t = t0

y1 = np.array([0 for \_ in range(N + 1)], dtype=np.double)

y2 = np.array([0 for \_ in range(N + 1)], dtype=np.double)

y1[0] = solution1(0)

y2[0] = solution2(0)

for i in range(N):

k1 = f1(t, y1[i], y2[i])

l1 = f2(t, y1[i], y2[i])

k2 = f1(t + c2 \* h, y1[i] + a21 \* h \* k1, y2[i] + a21 \* h \* l1)

l2 = f2(t + c2 \* h, y1[i] + a21 \* h \* k1, y2[i] + a21 \* h \* l1)

k3 = f1(t + c3 \* h, y1[i] + a31 \* h \* k1 + a32 \* k2 \* h, y2[i] + a31 \* h \* l1 + a32 \* l2 \* h)

l3 = f2(t + c3 \* h, y1[i] + a31 \* h \* k1 + a32 \* k2 \* h, y2[i] + a31 \* h \* l1 + a32 \* l2 \* h)

y1[i + 1] = y1[i] + h \* (b1 \* k1 + b2 \* k2 + b3 \* k3)

y2[i + 1] = y2[i] + h \* (b1 \* l1 + b2 \* l2 + b3 \* l3)

t = t + h

return y1, y2

def plot\_accuracy(ts, u, y, h):

plt.title('График точности приближения h= ' + str(h))

plt.xlabel('t') # устанавливаем метку оси х

plt.ylabel('u(t)') # устанавливаем метку оси у

plt.plot(ts, u, color='blue', label='u(t)') # передаем данные

plt.plot(ts, y, color='green', label='y(t)') # передаем данные

plt.legend()

plt.show()

def graph\_err(t0, T, N):

h = (T - t0) / N

ts = np.arange(t0, T + h, h)

u1, u2 = initial\_solution\_y(t0, T, N)

y1, y2 = method\_runge\_kutta(t0, T, N)

plot\_accuracy(ts, u1, y1, h)

plot\_accuracy(ts, u2, y2, h)

def plot\_max\_error(h, err):

plt.title('Зависимость максимальной погрешности от h ')

plt.xlabel('h') # устанавливаем метку оси х

plt.ylabel('e') # устанавливаем метку оси у

plt.plot(h, err) # передаем данные

plt.show() # рисуем график(должен появиться в отдельном окне)

def graph\_err\_from\_h(t0, T, Ns):

h = np.array([(T - t0) / N for N in Ns])

err1 = []

err2 = []

for N in Ns:

u1, u2 = initial\_solution\_y(t0, T, N)

y1, y2 = method\_runge\_kutta(t0, T, N)

err1.append(count\_error(y1, u1))

err2.append(count\_error(y2, u2))

plot\_max\_error(h, err1)

plot\_max\_error(h, err2)

def plot\_dependency(h, err):

plt.title('Зависимость e/h^3 от h')

plt.xlabel('h') # устанавливаем метку оси х

plt.ylabel('e') # устанавливаем метку оси у

plt.plot(h, err) # передаем данные

plt.show() # рисуем график(должен появиться в отдельном окне)

def graph\_err\_from\_h3(t0, T, Ns):

h = np.array([(T - t0) / N for N in Ns])

err1 = []

err2 = []

for N in Ns:

u1, u2 = initial\_solution\_y(t0, T, N)

y1, y2 = method\_runge\_kutta(t0, T, N)

r = count\_error(y1, u1)

p = count\_error(y2, u2)

h1 = (T - t0) / N

err1.append(r / h1 / h1 / h1)

err2.append(p / h1 / h1 / h1)

plot\_dependency(h, err1)

plot\_dependency(h, err2)

def main():

t0 = 0

T = 5

Ns = np.arange(1000, 5000, 50)

graph\_err\_from\_h(t0, T, Ns)

graph\_err\_from\_h3(t0, T, Ns)

graph\_err(t0, T, Ns[0])

if \_\_name\_\_ == '\_\_main\_\_':

main()

ecranoplan.py

import numpy as np

import matplotlib.pyplot as plt

P = 5000

S = 20

Iz = 125000

am = -1

l = 2

g = 9.81

ro = 1.25

aalpha = 5

aH = 2.05

aV = 0.005

c2 = 1 / 3

a21 = 1 / 3

c3 = 2 / 3

b2 = 0

b1 = 1 / 4

b3 = 3 / 4

a31 = 0

a32 = 2 / 3

V0 = 50

teta0 = 0.1

H0 = 0

F0 = 0

vsmall0 = 0.2

def nx(V):

return -1 \* aV \* ro \* S \* V \* V / (2 \* P)

def ny(alpha, V, H):

return (aalpha \* alpha - aH \* H) \* ro \* V \* V \* S / (2 \* P)

def Mz(alpha, V, l):

return am \* l \* alpha \* ro \* V \* V \* S / 2

def f1(V, teta): # V

return g \* (nx(V) - np.sin(teta))

def f2(V, teta, vsmall, H): # teta

return g \* (ny(vsmall - teta, V, H) - np.cos(teta)) / V

def f3(F): # vsmall

return F

def f4(V, teta, vsmall, l): # F

return Mz(vsmall - teta, V, l) / Iz

def f5(V, teta): # H

return V \* np.sin(teta)

def method\_runge\_kutta(t0, T, N, V0, l):

h = (T - t0) / N

t = t0

V = np.array([0 for i in range(N + 1)], dtype=np.double)

teta = np.array([0 for i in range(N + 1)], dtype=np.double)

vsmall = np.array([0 for i in range(N + 1)], dtype=np.double)

F = np.array([0 for i in range(N + 1)], dtype=np.double)

H = np.array([0 for i in range(N + 1)], dtype=np.double)

V[0] = V0

teta[0] = teta0

vsmall[0] = vsmall0

F[0] = F0

H[0] = H0

for i in range(N):

k1 = f1(V[i], teta[i]) # V

l1 = f2(V[i], teta[i], vsmall[i], H[i]) # teta

m1 = f3(F[i]) # vsmall

n1 = f4(V[i], teta[i], vsmall[i], l) # F

p1 = f5(V[i], teta[i]) # H

k2 = f1(V[i] + a21 \* h \* k1, teta[i] + a21 \* h \* l1)

l2 = f2(V[i] + a21 \* h \* k1, teta[i] + a21 \* h \* l1, vsmall[i] + a21 \* h \* m1, H[i] + a21 \* h \* p1)

m2 = f3(F[i] + a21 \* h \* n1)

n2 = f4(V[i] + a21 \* h \* k1, teta[i] + a21 \* h \* l1, vsmall[i] + a21 \* h \* m1, l)

p2 = f5(V[i] + a21 \* h \* k1, teta[i] + a21 \* h \* l1)

k3 = f1(V[i] + a32 \* k2 \* h, teta[i] + a32 \* l2 \* h)

l3 = f2(V[i] + a32 \* k2 \* h, teta[i] + a32 \* l2 \* h, vsmall[i] + a32 \* m2 \* h, H[i] + a32 \* p2 \* h)

m3 = f3(F[i] + a32 \* n2 \* h)

n3 = f4(V[i] + a32 \* k2 \* h, teta[i] + a32 \* l2 \* h, vsmall[i] + a32 \* m2 \* h, l)

p3 = f5(V[i] + a32 \* k2 \* h, teta[i] + a32 \* l2 \* h)

V[i + 1] = V[i] + h \* (b1 \* k1 + b2 \* k2 + b3 \* k3)

teta[i + 1] = teta[i] + h \* (b1 \* l1 + b2 \* l2 + b3 \* l3)

vsmall[i + 1] = vsmall[i] + h \* (b1 \* m1 + b2 \* m2 + b3 \* m3)

F[i + 1] = F[i] + h \* (b1 \* n1 + b2 \* n2 + b3 \* n3)

H[i + 1] = H[i] + h \* (b1 \* p1 + b2 \* p2 + b3 \* p3)

t = t + h

return V, teta, vsmall, F, H

t0 = 0

T = 20

N = 100000

def task4():

def show(xlabel, ylabel, x, y, title):

plt.xlabel(xlabel)

plt.ylabel(ylabel)

plt.title(title)

plt.grid(which='major')

plt.plot(x, y) # передаем данные

plt.show()

k = method\_runge\_kutta(t0, T, N, V0, l)

n = np.arange(t0, T + (T - t0) / N, (T - t0) / N) # устанавливаем метку оси у

show('t', 'V', n, k[0], 'Зависимость модуля скорости полета V от времени')

show('t', 'teta', n, k[1], 'Зависимость угла наклона траектории teta от времени')

show('t', 'v', n, k[2], 'Зависимость угла тангажа v от времени')

show('t', 'v\'', n, k[3], 'Зависимость производной угла тангажа v\' от времени')

show('t', 'H', n, k[4], 'Зависимость отклонения высоты полета от заданной H от времени')

def task5():

ls = np.arange(2, 10, 2)

v0s = np.arange(50, 100, 10)

dictl = {0: [], 1: [], 2: [], 3: [], 4: []}

dictv = {0: [], 1: [], 2: [], 3: [], 4: []}

def show(ary, x, title, k, ls, labelstr, ylabel):

plt.title(title)

plt.xlabel('t')

plt.ylabel(ylabel)

for i in range(k):

plt.plot(x, ary[i], label=labelstr + str(ls[i]))

plt.legend()

plt.grid(which='major')

plt.show()

for l1 in ls:

k = method\_runge\_kutta(t0, T, N, V0, l1)

for i in range(5):

dictl[i].append(k[i])

for vv0 in v0s:

k = method\_runge\_kutta(t0, T, N, vv0, l)

for i in range(5):

dictv[i].append(k[i])

titles = ['График модуля скорости полета V от времени', 'График угла наклона траектории teta от времени',

'График угла тангажа v от времени',

'График производной угла тангажа v\' от времени',

'График отклонения высоты полета от заданной H от времени']

n = np.arange(t0, T + (T - t0) / N, (T - t0) / N) # устанавливаем метку оси у

labels = ['V', 'teta', 'v', 'v\'', 'H']

for i in range(5):

show(dictl[i], n, titles[i], len(ls), ls, 'при l = ', labels[i])

for i in range(5):

show(dictv[i], n, titles[i], len(v0s), v0s, 'при v = ', labels[i])

def main():

task4()

task5()

if \_\_name\_\_ == '\_\_main\_\_':

main()