Московский авиационный институт

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)  
–––––––––––––––  
Аэрокосмический факультет   
––––––––––––––––  
кафедра 611Б «системный анализ и управление»

Курсовая работа

по дисциплине: «Динамика летательных аппаратов»

на тему: «прогнозирование изменения параметров орбиты ИСЗ под действием возмущающих факторов»

выполнил:

студент группы 60-307Б

Савенков В.Р.,

проверила:

Морозова Т.А.

Москва 2021г

Исходные данные

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i, град | ha, км | hп, км | 𝛀, град | U, град | Sa, м2 | m,  кг | Cxa | Fa, Вт/м2 | Возмущающий фактор |
| 20.8 | 1740 | 350 | 10 | 7 | 23 | 1700 | 3.5 | 125 | Нецентральность гравитационного поля |

**Цель работы**

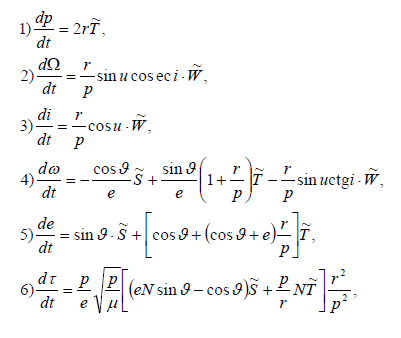
Разработать программу, которая даст возможность прогнозировать изменение параметров орбиты ИСЗ под действием возмущающих факторов.

**Решение задачи.**

Для построения модели возмущенного движения используется метод оскулирующих элементов.

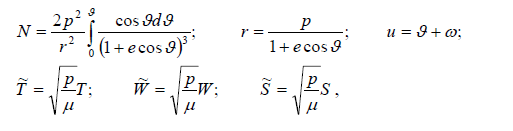
Метод разработан Лагранжем и получил широкое распространение при исследовании возмущенного движения. Сущность метода заключается в том, что возмущенную (истинную) траекторию КА рассматривают как состоящую из последовательности невозмущенных траекторий с разными параметрами для каждого текущего момента времени. В итоге траектория возмущенного движения в каждый момент времени соприкасается с траекторией невозмущенного движения для этого же момента времени и представляет собой огибающую семейства невозмущенных траекторий движения.

В результате имеем следующую систему уравнений

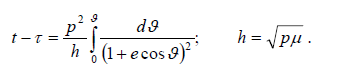


(1)

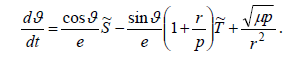
Где



а истинная аномалия θ связана со временем t формулой

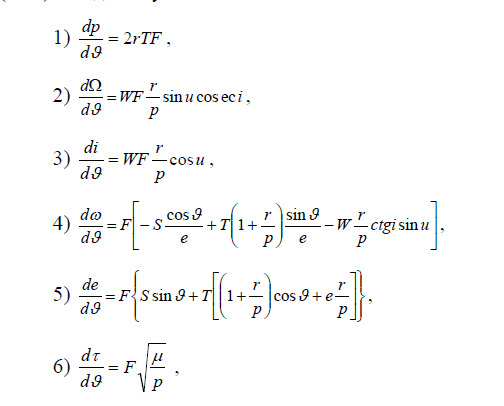


На величины возмущающих ускорений *S,T,W*, входящих в систему уравнений не наложено никаких ограничений. Если возмущающие ускорения *S,T,W* не зависят явно от времени *t*, то правые части уравнений будут функциями только от истинной аномалии и элементов орбиты. Поэтому в системе можно исключить время, приняв за независимую переменную θ . Можно показать, что связь между θ и t определяется соотношением:



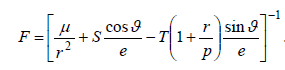
(2)

Каждое из уравнений системы (1) разделим почленно на уравнение (2). Тогда получим:



(3)

Где

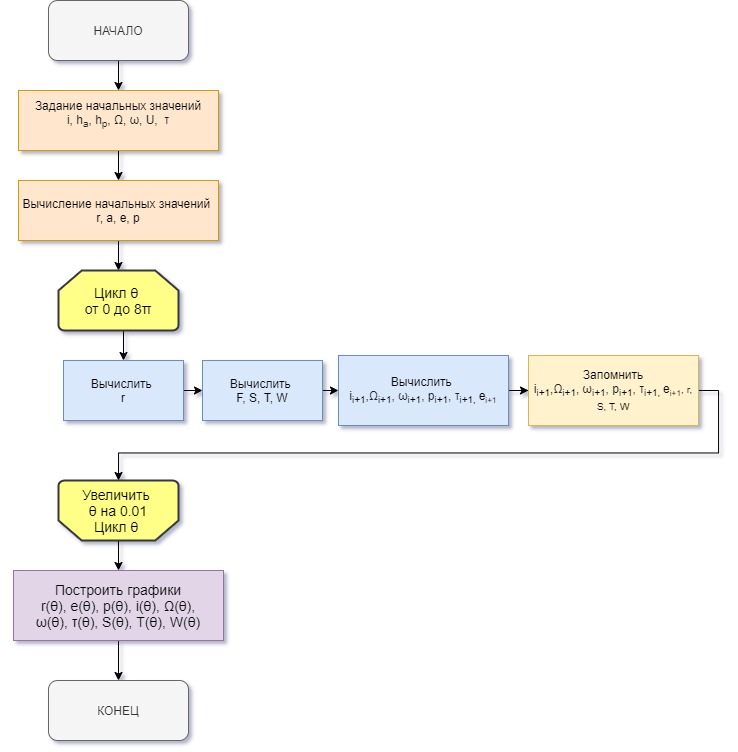


(4)

Для решения системы дифференциальных уравнений (3) используется метод Эйлера. Приближенное решение в узлах xi, которое обозначим через yi, определяется по формуле:

Для разработки программы используется язык программирования Python 3.9 и библиотека MATPLOTLIB для визуализации полученных результатов.

Общий алгоритм решения поставленной задачи представлен на блок-схеме 1.



Блок-схема 1

**Анализ работы:**

Орбита ИСЗ в полярной системе координат с учетом воздействия внешних сил на параметры орбиты. Результаты исследований представить в виде графиков для следующих элементов орбиты: фокальный параметр, эксцентриситет, аргумент перицентра, долгота восходящего узла, наклонение орбиты и трех составляющих возмущающего ускорения (трансверсальная, радиальная и нормальная).

В результате выполнения алгоритма была получена Орбита ИСЗ в полярной системе координат с учетом воздействия внешних сил на параметры орбиты.

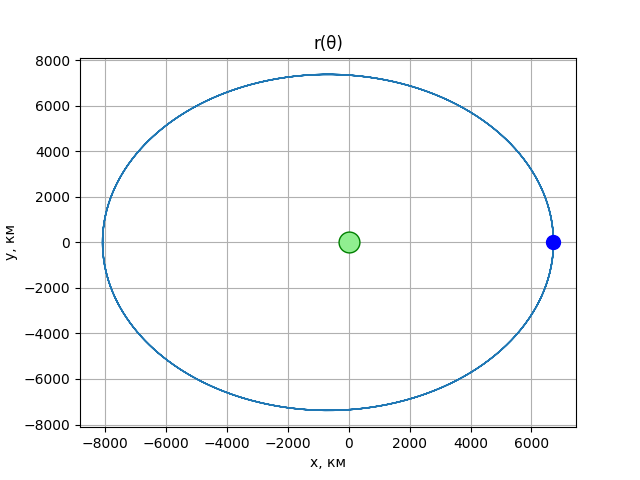


Рис 1. Орбита движения ИСЗ с учетом воздействия внешних сил на параметры орбиты

Также были получены зависиости от значения истинной аномалии для фокального параметр, эксцентриситета, аргумента перицентра, долготы восходящего узла, наклонения орбиты и трех составляющих возмущающего ускорения.

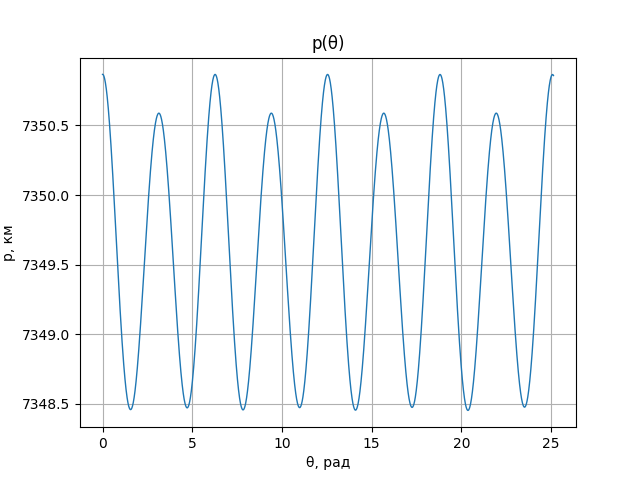


Рис 2. Зависимость фокального параметра от истинной аномалии.

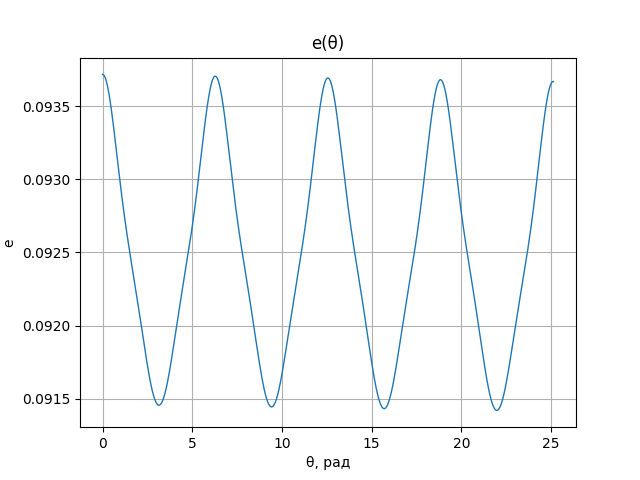


Рис 3. Зависимость эксцентриситета орбиты от истинной аномалии.

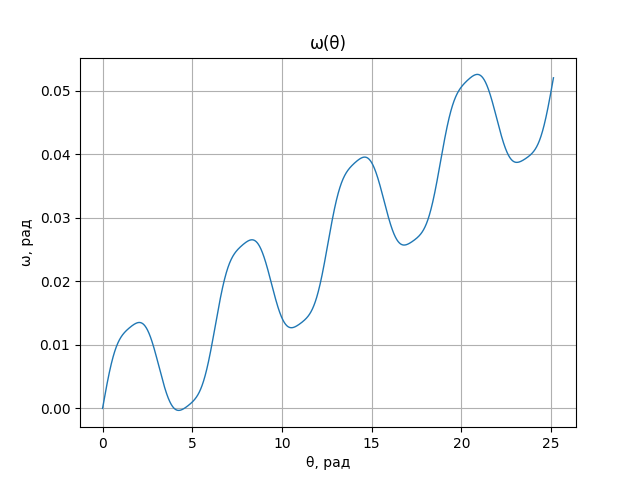


Рис 4. Зависимость аргумента перицентра от истинной аномалии.

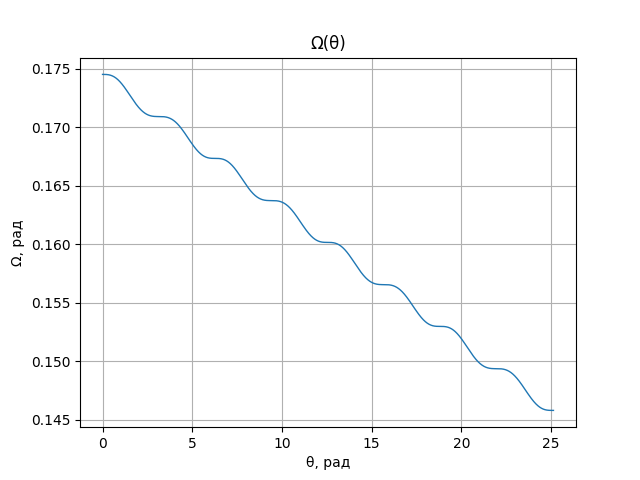


Рис 5. Зависимость долготы восходящего узла от истинной аномалии.

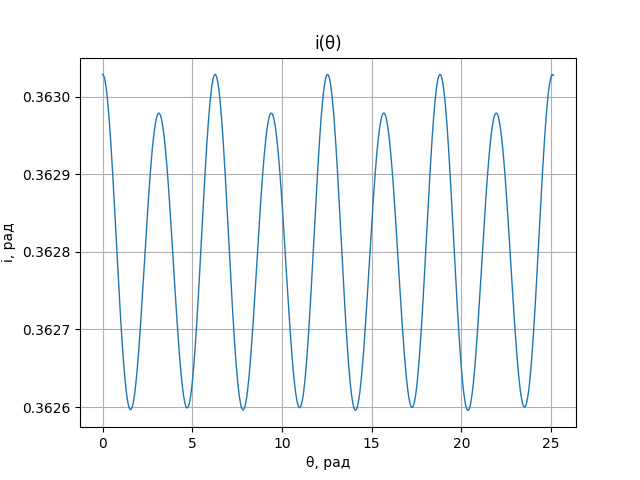


Рис 6. Зависимость наклонения от истинной аномалии.

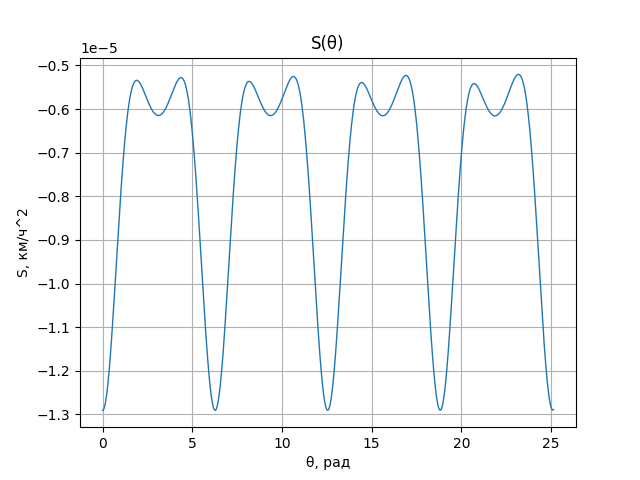


Рис 7. Зависимость трансверсальной составляющей возмущающего ускорения от истинной аномалии.

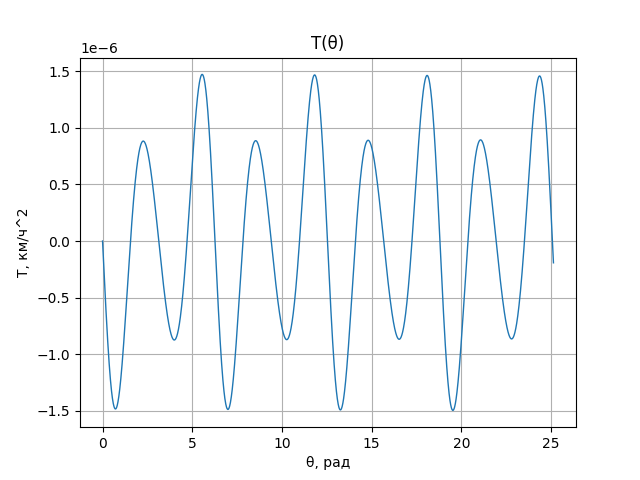


Рис 8. Зависимость радиальной составляющей возмущающего ускорения от истинной аномалии.

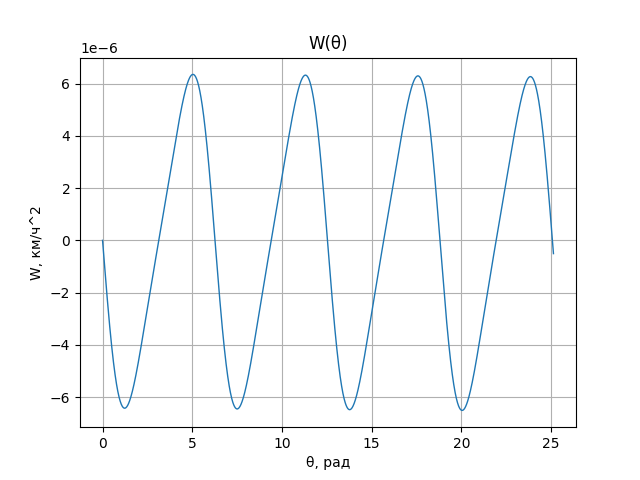


Рис 9. Зависимость нормальной составляющей возмущающего ускорения от истинной аномалии.

Вывод: 1111111

Список литературы

1. А. С. Шалыгин, В. А. Санников, И.Л.Петрова «Баллистика космических аппаратов»
2. Д. Г. Васильев, В. В. Бетанов «Применение методом имитационного моделирования в задачах околоземных космических аппаратов»
3. А. В. Ващенко «Влияние возмущающих гравитационных сил, связанных с нецентральностью гравитационного поля Земли, на эволюцию орбиты космического аппарата»

**Приложение**

Исходные коды.

**data.py**

**import** **math**

# 11 ВАРИАНТ

# ИСХОДНЫЕ ДАННЫЕ

# ======================

i = math.radians(**20.8**)

h\_a = **1740**

h\_p = **350**

OMEGA = math.radians(**10**)

U = math.radians(**7**)

Sa = **23**

m = **1700**

C\_xa = **3.5**

Fa = **125**

# ======================

d\_theta = **1e-2**

Earth\_radius = **6371**

**functions.py**

**import** **math**

# Формулы для рассчёта компонентов возмущающего ускорения,

# вызванного нецентральность гравитационного поля Земли

# ===================================================================================

Epsilon = **398603** \* **66.07** \* **1e3**

S1 = **lambda** r, i, u: Epsilon / r \*\* **4** \* (**3** \* math.sin(i) \*\* **2** \* math.sin(u) \*\* **2** - **1**)

T1 = **lambda** r, i, u: -Epsilon / r \*\* **4** \* math.sin(i) \*\* **2** \* math.sin(**2** \* u)

W1 = **lambda** r, i, u: -Epsilon / r \*\* **4** \* math.sin(**2** \* i) \* math.sin(u)

# ===================================================================================

# Правые части диф уравнений метода оскулирующих элементов

# =====================================================================================================================

R\_p = **lambda** r, T, F: **2** \* r \* T \* F

R\_OMEGA = **lambda** W, F, r, p, u, i: W \* F \* r / p \* math.sin(u) / math.sin(i)

R\_i = **lambda** W, F, r, p, u: W \* F \* r / p \* math.cos(u)

R\_omega = **lambda** F, S, theta, e, T, r, p, W, i, u: F \* (

-S \* math.cos(theta) / e + T \* (**1** + r / p) \* math.sin(theta) / e - W \* r / p \* (**1** / math.tan(i)) \* math.sin(u))

R\_e = **lambda** F, S, theta, T, r, p, e: F \* (S \* math.sin(theta) + T \* ((**1** + r / p) \* math.cos(theta) + e \* r / p))

R\_tau = **lambda** F, p: F \* math.sqrt(**398603** / p)

geT\_F = **lambda** r, S, T, theta, e, p: (**398603** / r \*\* **2** + S \* math.cos(theta) / e - T \* (**1** + r / p) \* math.sin(

theta) / e) \*\* -**1**

# =====================================================================================================================

# Вспомогательные функции

# ========================================================

get\_a = **lambda** ra, rp: **0.5** \* (ra + rp)

get\_e = **lambda** ra, rp, a: (ra - rp) / (**2** \* a)

get\_r = **lambda** p, e, theta: p / (**1** + e \* math.cos(theta))

get\_p = **lambda** a, e: a \* (**1** - e \*\* **2**)

# ========================================================

**main.py**

**from** **data** **import** \*

**from** **functions** **import** \*

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

revolutions = **8** # Количество оборотов

theta\_list = list(np.arange(**0**, revolutions \* math.pi + d\_theta, d\_theta))

r\_list = []

p\_list = []

OMEGA\_list = []

i\_list = []

omega\_list = []

e\_list = []

tau\_list = []

S\_list = []

T\_list = []

W\_list = []

# НАЧАЛЬНЫЕ ЗНАЧЕНИЯ

# ==========================================================

a = get\_a(ra=Earth\_radius + h\_a, rp=Earth\_radius + h\_p)

e = get\_e(ra=Earth\_radius + h\_a, rp=Earth\_radius + h\_p, a=a)

p = get\_p(a=a, e=e)

omega = **0**

tau = **0**

r = get\_r(p=p, e=e, theta=theta\_list[**0**])

# ==========================================================

p\_list.append(p)

OMEGA\_list.append(OMEGA)

omega\_list.append(omega)

i\_list.append(i)

e\_list.append(e)

tau\_list.append(tau)

**def** **iterative\_loop**():

**global** r, p, OMEGA, omega, i, e, tau

r1 = p1 = OMEGA1 = omega1 = i1 = e1 = tau1 = **0**

**for** theta **in** theta\_list:

# Компоненты возмущающих ускорений

# ===============================

T = T1(r=r, i=i, u=theta + omega)

S = S1(r=r, i=i, u=theta + omega)

W = W1(r=r, i=i, u=theta + omega)

# T = S = W = 0

S\_list.append(S)

T\_list.append(T)

W\_list.append(W)

# ===============================

# Радиус

# ==============================

r = get\_r(p=p, e=e, theta=theta)

r\_list.append(r)

# ==============================

F = geT\_F(r=r, S=S, T=T, theta=theta, e=e, p=p)

# Новые элементы

# ======================================================================================================

p1 = p + R\_p(r=r, T=T, F=F) \* d\_theta

OMEGA1 = OMEGA + R\_OMEGA(W=W, F=F, r=r, p=p, u=theta + omega, i=i) \* d\_theta

i1 = i + R\_i(W=W, F=F, r=r, p=p, u=theta + omega) \* d\_theta

omega1 = omega + R\_omega(F=F, S=S, theta=theta, e=e, T=T, r=r, p=p, W=W, i=i, u=theta + omega) \* d\_theta

e1 = e + R\_e(F=F, S=S, theta=theta, T=T, r=r, p=p, e=e) \* d\_theta

tau1 = tau + R\_tau(F=F, p=p) \* d\_theta

# ======================================================================================================

# Занесение новых данных в контейнеры

# ===================================

p\_list.append(p1)

OMEGA\_list.append(OMEGA1)

i\_list.append(i1)

omega\_list.append(omega1)

e\_list.append(e1)

tau\_list.append(tau1)

# ===================================

p, i, e, tau, OMEGA, omega = p1, i1, e1, tau1, OMEGA1, omega1

**def** **chart**(x, y, title='', xlabel='', ylabel=''):

plt.plot(x, y, linewidth=**1**)

**if** title == 'r(θ)':

plt.plot(**0**, **0**, marker='.', markersize=**30**, color='green', markerfacecolor='lightgreen')

plt.plot(x[**0**], y[**0**], marker='.', markersize=**20**, color='blue', markerfacecolor='blue')

plt.xlabel(xlabel)

plt.ylabel(ylabel)

plt.title(title)

plt.grid(True)

**if** title == 'ω(θ)':

title = 'ω (θ)'

plt.savefig('./charts/{}.png'.format(title))

plt.show()

**def** **main**():

iterative\_loop()

x = [r\_list[i] \* math.cos(theta\_list[i]) **for** i **in** range(len(theta\_list))]

y = [r\_list[i] \* math.sin(theta\_list[i]) **for** i **in** range(len(theta\_list))]

chart(x=x, y=y, title='r(θ)', xlabel='x, км', ylabel='y, км')

chart(x=theta\_list, y=e\_list[:len(theta\_list)], title='e(θ)', xlabel='θ, рад', ylabel='e')

chart(x=theta\_list, y=i\_list[:len(theta\_list)], title='i(θ)', xlabel='θ, рад', ylabel='i, рад')

chart(x=theta\_list, y=p\_list[:len(theta\_list)], title='p(θ)', xlabel='θ, рад', ylabel='p, км')

chart(x=theta\_list, y=OMEGA\_list[:len(theta\_list)], title='Ω(θ)', xlabel='θ, рад', ylabel='Ω, рад')

chart(x=theta\_list, y=omega\_list[:len(theta\_list)], title='ω(θ)', xlabel='θ, рад', ylabel='ω, рад')

chart(x=theta\_list, y=tau\_list[:len(theta\_list)], title='τ(θ)', xlabel='θ, рад', ylabel='τ, час')

chart(x=theta\_list, y=S\_list[:len(theta\_list)], title='S(θ)', xlabel='θ, рад', ylabel='S, км/ч^2')

chart(x=theta\_list, y=T\_list[:len(theta\_list)], title='T(θ)', xlabel='θ, рад', ylabel='T, км/ч^2')

chart(x=theta\_list, y=W\_list[:len(theta\_list)], title='W(θ)', xlabel='θ, рад', ylabel='W, км/ч^2')

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

main()