Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский университет
«Московский институт электронной техники»

Кафедра высшей математики №1

Стадник Александр Михайлович

Лабораторная работа № 1
по теме «Этапы построения математической модели»
Направленность (профиль) «Применение математических методов к решению инженерных и естественнонаучных задач»

Задача о футболисте

Студент _____ Стадник А.М

Москва 2022

Объект исследования задачи

Задача

1. Содержательная постановка задачи

Разработать математическую модель, позволяющую описать удар футбольного мяча ворота. Модель должна:

- Сформулировать условие попадания мяча в створ ворот
- Найти на поле область, из которой игрок попадет в ворота соперника при заданном угле удара и скорости удара, изменяя угол между линией ворот и плоскостью удара.

Исходные данные:

- Скорость мяча при ударе V_0
- Угол удара мяча α
- Координаты ворот $(gateX,\ gateYUpper)$, $(gateX,\ gateYMiddle,\ gateYButtom)$
- Длина и ширина поля $(fieldLength, fieldWidth)$
- Ширина и высота ворот $(gateWidth, gateHeight)$

2. Концептуальная постановка задачи

Движение футбольного мяча может быть описана в соответствии с законом классической механики Ньютона. Применим следующие гипотезы:

- Объектом моделирования является футбольный мяч, который мы принимаем за материальную точку.
- Движение происходит в поле силы тяжести с постоянным ускорением свободного падения g.
- Движения мяча происходит в одной области, перпендикулярно поверхности земли и проходящий через точку середины ворот.
- Пренебрегаем сопротивлением воздуха и возмущениями, вызванными собственным вращением мяча вокруг центра масс.
- Ворота располагаются ровно по середине короткой кромки поля.

На основе гипотез имеем следующие выводы:

- В качестве параметров движения мяча можем использовать его координаты $x, y\ z$ и соответствующие проекции скорости V_x, V_y, V_z. Для определения положения мяча нужно найти зависимость этих координат от времени t.
- Попадание мяча в ворота означает, что он находится в пределах ширины и высоты ворот при пересечении их плоскости.

Сокращенная формулировка задачи концептуальной постановки задачи:

- Определить закон движения материальной точки при заданных координатах начала движения, начальной скорости и направлении начала движения. Вычислить координаты мяча при пересечении плоскости ворот и определить, входят ли они в пределы ворот.

3. Математическая постановка задачи

Угол β вычисляется, как угол между плоскостью перпендикулярной поверхности земли и проходящий через точку середины ворот и поверхности земли и проходящий через верхний или нижний край ворот (нижний, если координата начала удара по ширине меньше координаты центра ворот и верхней, если больше)

Проекции скорости на основании заданных углов удара:

$V_z = V_0 * \sin(\alpha)$

$V_x = V_0 * \cos(\beta) * \cos(\alpha)$

$V_y = V_0 * \cos(\beta) * \sin(\alpha)$

Координаты мяча в момент времени t:

$y_t = playerY + V_y * t$

$x_t = playerX + V_x * t$

$z_t = V_z * t - \frac{gt^2}{2}$

4. Качественный анализ и проверка конкретности модели

Контроль размерности:

- Скорость и её проекции имеют размерность $\frac{\text{м}}{\text{с}}$, координаты - м, время - с, ускорение свободного падения - $\frac{\text{м}}{\text{с}^2}$. При таких размерностях все используемые формулы работают корректно.

5. Выбор и обоснование методов решения

Аналитический (численный) метод

```
from numpy import *
from matplotlib.patches import *
import matplotlib.pyplot as plt
import math
import numpy as np

V0 = (float(input("Введите скорость мяча после удара (м/с): ")))
alpha0 = math.radians(
    float(input("Введите угол, под которым был ударен мяч (градусы): ")))



g = 9.8
fieldLength, fieldWidth = 105, 68,
gateHeight, gateWidth = 2.44, 7.32



gateX = fieldLength
gateYMiddle, gateYBottom, gateYUpper = fieldWidth / 2, fieldWidth / 2 - \
    gateWidth / 2, fieldWidth / 2 + gateWidth / 2

gateUpperCords = np.array([gateX, gateYUpper])
gateMiddleCords = np.array([gateX, gateYMiddle])
gateBottomCords = np.array([gateX, gateYBottom])

gateButtomCords = np.array([gateX, gateYBottom])
```

```
gateUpperCords = np.array([gateX, gateYUpper])
maxLength = math.floor((V0**2 * math.sin(2 * alpha0)) / g)
maxTime = 2 * V0 * math.sin(alpha0) / g

print("V0: ", V0)
print("maxLength: ", maxLength)
print("math.sin(alpha0): ", math.sin(alpha0))
print("maxTime: ", maxTime)



def rotate(origin, point, angle):

    R = np.array([[np.cos(angle), -np.sin(angle)],
                  [np.sin(angle),  np.cos(angle)]])
    o = np.atleast_2d(origin)
    point = np.atleast_2d(point)
    return np.squeeze((R @ (point.T-o.T) + o.T).T)



def dotproduct(v1, v2):
    iter = ((a * b) for a, b in zip(v1, v2))
    return np.sum(np.fromiter(iter, float))



def length(v):
    return math.sqrt(dotproduct(v, v))
```

```
        def angleBetween(v1, v2):
            if length(v1) * length(v2) == float(0):
                return 0
            return math.acos(dotproduct(v1, v2) / (length(v1) * length(v2)))


        def checkIntersection(A, B, C):
            return (C[1] - A[1]) * (B[0] - A[0]) > (B[1] - A[1]) * (C[0] - A[0])


        def isIntersect(A, B, C, D):
            return checkIntersection(A, C, D) != checkIntersection(B, C, D) and
checkIntersection(A, B, C) != checkIntersection(A, B, D)


        def isScored(playerX, playerY):
            playerCords = np.array([playerX, playerY])
            goalCordsWithoutRotate = np.array([playerX + maxLength, playerY])

            angle = angleBetween(np.array(gateMiddleCords - playerCords),
                                 np.array(goalCordsWithoutRotate - playerCords),)

            if playerY > fieldWidth / 2:
                angle = -angle

            goalCords = rotate(playerCords, goalCordsWithoutRotate, angle)

            if (isIntersect(playerCords, goalCords, gateBottomCords, gateUpperCords)):
```

```
            timeToGate = (fieldLength - playerX) / \
                (V0 * math.cos(angle) * math.cos(alpha0))

            if timeToGate > maxTime:
                timeToGate = 0

            isHop = V0 * math.sin(alpha0) * timeToGate - \
                (g * timeToGate**2) / 2 > gateHeight

            if not isHop:
                return True

            return False

        return False


    fig = plt.figure()
    fig.set_size_inches(7, 5)
    ax = fig.add_subplot(1, 1, 1)

    Pitch = Rectangle([0, 0], width=fieldLength, height=fieldWidth, fill=False)

    Gate = Rectangle(
        [fieldLength - 0.5, fieldWidth / 2 - gateWidth / 2], width=1,
height=gateWidth, fill=True)
```

```
element = [Pitch, Gate]
for i in element:
    ax.add_patch(i)


plt.ylim(-2, 82)
plt.xlim(-2, 122)
plt.axis('off')


print('Вычисляем...')
for playerX in range(fieldLength + 1):
    for playerY in range(fieldWidth + 1):
        if isScored(playerX, playerY):
            ax.add_patch(plt.Circle((playerX, playerY), 0.6, color="black"))


print('БУМ!')
plt.show()
```

6. Проверка адекватности модели

Данная математическая модель для решения поставленной задачи адекватна. Модель может быть усовершенствована путем добавления дополнительных параметров объектов, таких как вращение и инерция. Данную модель можно применять для примерно обнаружения области попадания мяча при заданных параметрах.

7. Практическое использование построенной модели

Пример работы программы:

```
(venv) C:\Users\Aleksandr\Desktop\Study\7_sem\matmod\lab1>python lab1.py
Введите скорость мяча после удара (м/с): 20
Введите угол, под которым был ударен мяч (градусы): 30
Вычисляем...
Вычислено!
```

(Вид сверху; закрашенная область - область, из которой можно попасть в ворота заданным ударом, ворота обозначены красной линией):