Отсчет по лабораторной работе №3

Модель боевых действий

Динькиев Валерий

Содержание

# Цель работы

Построить упрощенную модель боевых действий с помощью Python.

# Задание

**Вариант 16** Между страной и страной идет война. Численности состава войск исчисляются от начала войны и являются временными функциями и . В начальный момент времени страна имеет армию численностью 39 800 человек, а в распоряжении страны армия численностью в 21 400 человек. Для упрощения модели считаем, что коэффициенты постоянны. Также считаем и непрерывными функциями.

Постройте графики изменения численности войск армии и армии для следующих случаев:

1. Модель боевых действий между регулярными войсками
2. Модель ведение боевых действий с участием регулярных войск и партизанских отрядов

# Выполнение лабораторной работы

**1. Модель боевых действий между регулярными войсками.**

1.1. Изучил начальные условия. Коэффициент смертности, не связанный с боевыми действиями, у первой армии 0,42, а у второй – 0,43. Коэффициент эффективности первой и второй армии 0,59 и 0,68 соответственно. Функция, описывающая подход подкрепление первой армии, , подкрепление второй армии описывается функцией . – людей 1-ой армии, – людей 2-ой армии.

1.2. Построим численное решение для начальных условий:

x\_0 = 39800  
y\_0 = 21400  
a\_1 = 0.42  
b\_1 = 0.68  
c\_1 = 0.59  
h\_1 = 0.43  
def P\_1(t):  
 p\_1 = np.sin(5t + 1)  
 return p\_1  
  
def Q\_1(t):  
 q\_1 = np.cos(5t + 2)  
 return q\_1

1.3. Для времени задал следующие условия: – начальный момент времени, –предельный момент времени, – шаг изменения времени.

1.4. Добавил в программу условия, описывающие время:

t\_0 = 0  
t\_max = 1  
dt = 0.05  
t = np.arange(t\_0, t\_max, dt)

1.5. Запрограммировал заданную систему дифференциальных уравнений, описывающих изменение количества людей в армии:

def S\_1(f, t):  
 s\_1 = -a\_1\*f[0] - b\_1\*f[1] + P\_1(t)  
 s\_2 = -c\_1\*f[0] - h\_1\*f[1] + Q\_1(t)  
 return s\_1, s\_2

1.6. Создал вектор начальной численности армий:

v = np.array([x0, y0])

1.7. Запрограммировал решение системы уравнений:

f\_1 = odeint(S1, v, t)

1.8. Построил график изменения численности армий:

plt.plot(t, f1)

2.1. Изучил начальные условия. Коэффициент смертности, не связанный с боевыми действиями, у первой армии 0,301, а у второй – 0,4. Коэффициент эффективности первой и второй армии 0,502 и 0,7 соответственно. Функция, описывающая подход подкрепление первой армии, , подкрепление второй армии описывается функцией . – людей 1-ой армии, – людей 2-ой армии.

2.2. Построим численное решение для начальных условий:

x\_0 = 39800  
y\_0 = 21400  
a\_2 = 0.301  
b\_2 = 0.7  
c\_2 = 0.502  
h\_2 = 0.4  
def P\_2(t):  
 p\_2 = np.sin(20t) + 1  
 return p\_2  
  
def Q\_2(t):  
 q\_2 = np.cos(20t) + 1  
 return q\_2

2.3. Для времени задал следующие условия: – начальный момент времени, –предельный момент времени, – шаг изменения времени.

2.4. Добавил в программу условия, описывающие время:

t\_0 = 0  
t\_max = 1  
dt = 0.05  
t = np.arange(t\_0, t\_max, dt)

2.5. Функция заданной системы дифференциальных уравнений, которое описывает изменение количества людей в армии:

def S\_2(f, t):  
 s\_1 = -a\_2\*f[0] - b\_2\*f[1] + P\_2(t)  
 s\_2 = -c\_2\*f[0] - h\_2\*f[1] + Q\_2(t)  
 return s\_1, s\_2

2.6. Запрограммировал решение системы уравнений:

f\_2 = odeint(S\_2, v, t)

2.7. Построил график изменения численности армий:

plt.plot(t, f\_2)

**3. Программа на Python**

import math  
import numpy as np  
from scipy.integrate import odeint  
import matplotlib.pyplot as plt  
x\_0 = 39800  
y\_0 = 21400  
  
a\_1 = 0.42  
b\_1 = 0.68  
c\_1 = 0.59  
h\_1 = 0.43  
  
a\_2 = 0.301  
b\_2 = 0.7  
c\_2 = 0.502  
h\_2 = 0.4  
  
t0 = 0  
tmax = 1  
dt = 0.05  
t = np.arange(t0, tmax, dt)  
  
def P\_1(t):  
 p\_1 = np.sin(5\*t + 1)  
 return p\_1  
  
def Q\_1(t):  
 q\_1 = np.cos(5\*t + 2)  
 return q\_1  
  
def P\_2(t):  
 p\_2 = np.sin(20\*t) + 1  
 return p\_2  
  
def Q\_2(t):  
 q\_2 = np.cos(20\*t) + 1  
 return q\_2  
  
def S\_1(f, t):  
 s\_1 = -a\_1\*f[0] - b\_1\*f[1] + P\_1(t)  
 s\_2 = -c\_1\*f[0] - h\_1\*f[1] + Q\_1(t)  
 return s\_1, s\_2  
  
def S\_2(f, t):  
 s\_1 = -a\_2\*f[0] - b\_2\*f[1] + P\_2(t)  
 s\_2 = -c\_2\*f[0] - h\_2\*f[1] + Q\_2(t)  
 return s\_1, s\_2  
  
v = np.array([x0, y0])  
f\_1 = odeint(S\_1, v, t)  
f\_2 = odeint(S\_2, v, t)  
  
plt.plot(t, f\_2)  
plt.ylabel('Количество людей в армии')  
plt.xlabel('Время')  
plt.legend(['Армия X', 'Армия Y'])  
  
plt.plot(t, f\_1)  
plt.ylabel('Количество людей в армии')  
plt.xlabel('Время')  
plt.legend(['Армия X', 'Армия Y'])

3.1. Получил графики изменения численностей армий (см. рис. 1 и 2)

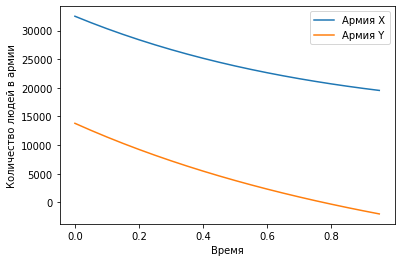


Figure 1: График боевых действий между регулярными войсками

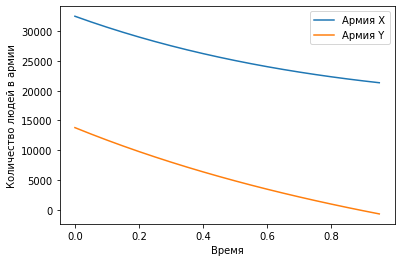


Figure 2: График боевых действий с участием регулярных войск и партизанских отрядов

# Выводы

Построил упрощенную модель боевых действий с помощью Python. В боевых действиях между регулярными войсками победит армия X. Также можно видеть по графику, что армии X понадобится довольно много времени, армию Y. В боевых действиях с участием регулярных войск и партизанских отрядов также победит армия Х, но с меньшими потерями чем в случае с регулярными войсками.