Лабораторная работа №3

Модель боевых действий - Модель Ланчкстера

Абу Сувейлим Мухаммед Мунифович

Содержание

Список иллюстраций

Цель работы

- Целью работы является познокомится с простейшими моделими боевых действий – модели Ланчестера.
- Сделать начальный аналих этих модлей.

Задание

1. Постройте графики изменения численности войск армии X и армии У для следующих случаев:
    - Модель боевых действий между регулярными войсками;
    - Модель боевых действий между регулярными войсками.
2. Графики должны быть созданы/построенны используя Julia и OpenModelica.

Теоретическое введение

Законы Ланчестера (законы Осипова — Ланчестера) — математическая формула для расчета относительных сил пары сражающихся сторон — подразделений вооруженных сил. В статье «Влияние численности сражающихся сторон на их потери», опубликованной журналом «Военный сборник» в 1915 году, генерал-майор Корпуса военных топографов М. П. Осипов описал математическую модель глобального вооружённого противостояния, практически применяемую в военном деле при описании убыли сражающихся сторон с течением времени и, входящую в математическую теорию исследования операций, на год опередив английского математика Ф. У. Ланчестера. Мировая война, две революции в России не позволили новой власти заявить в установленном в научной среде порядке об открытии царского офицера.

Уравнения Ланчестера — это дифференциальные уравнения, описывающие зависимость между силами сражающихся сторон A и D как функцию от времени, причем функция зависит только от A и D.

Выполнение лабораторной работы

Модель боевых действий между регулярными войсками

Модлеирование на языке программеровании Julia

1. Во-первых, я использвал пакеты Plots и DifferentialEquations.

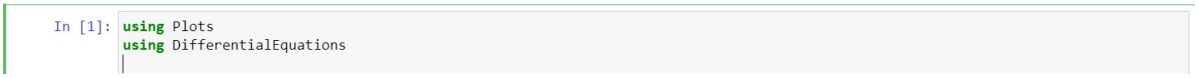

Рис. 1: Plots и DifferentialEquations Packges

2. Инициализировал нужны нам константи и функции в моделии. x0 - численность первой армии x; y0 - численность второй армии y; a - константа, характеризующая степень влияния различных факторов на потери; b - эффективность боевых действий армии y; c - ффективность боевых действий армии x; h - константа, характеризующая степень влияния различных факторов на потери. P(t)- возможность подхода подкрепления к армии x; Q(t) - возможность подхода подкрепления к армии y.

```julia
x0 = 22022 #численность первой армии x
y0 = 33033 #численность второй армии y

a = 0.401 #константа, характеризующая степень влияния различных факторов на потери
b = 0.707 #эффективность боевых действий армии y
c = 0.606 #ффективность боевых действий армии x
h = 0.502 # константа, характеризующая степень влияния различных факторов на потери

P(t) = sin(8*t) #возможность подхода подкрепления к армии x
Q(t) = cos(6*t) #возможность подхода подкрепления к армии y
```

Рис. 2: Initial Values

3. Я еше добавил интервал времении от 0 до 1.

```julia
manpower = [x0, y0]
p = (a, b, c, h)
T = [0, 1]
```

Рис. 3: Time interval for simulation

4. Теперь можно построиет модель боевых действий между регулярными войсками №1

```julia
function modelOne(du, u, p, t)
    a, b, c, h = p
    du[1] = -a * u[1] - b * u[2] + P(t)
    du[2] = -c * u[1] - b * u[2] + Q(t)
end

prob = ODEProblem(modelOne, manpower, T, p)
```

Рис. 4: Model One

5. Осталось только решить ОДУ.

```
In [2]: solOne = solve(prob)

Out[2]: retcode: Success
        Interpolation: specialized 4th order "free" interpolation, specialized 2nd order "free" stiffness-aware interpolation
        t: 6-element Vector{Float64}:
         0.0
         0.0912317087351149
         0.29374139136324146
         0.5487791707593012
         0.8558401350298304
         1.0
        u: 6-element Vector{Vector{Float64}}:
         [22022.0, 33033.0]
         [19241.539558581208, 29867.20556092709]
         [14059.625394370096, 23998.30532335177]
         [9090.096248812577, 18430.046584664873]
         [4795.610903896066, 13706.681303525214]
         [3254.4482918580984, 12047.185266213253]
```

Рис. 5: ODE

6. График уоказывет числоности армии X и Y. Армия Y побеждает.

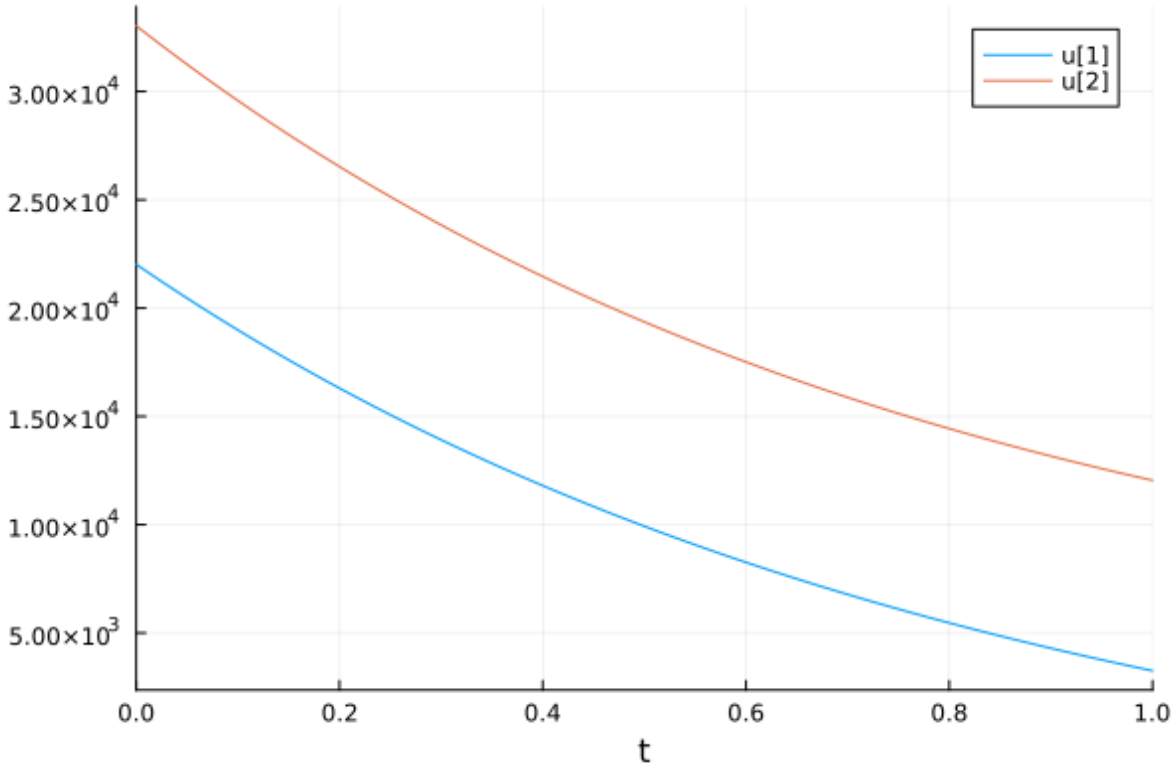

Рис. 6: Model One Graph

Модлеирование на языке программировании OpenModelica

1. В OpenModelica все прощее. Я просто переписал код из Julia. В этой прошраиие все величины имею тот же смысл, что и в Julia.

```
model lab3_1

Real x;
Real y;

Real a = 0.401;
Real b = 0.707;
Real c = 0.606;
Real h = 0.502;
Real t = time;

initial equation
x = 22022;
y = 33033;

equation
der(x) = -a*x - b*y + sin(8*t);
der(y) = -c*x -b*y + cos(6*t);

end lab3_1;
```

Рис. 7: OpenModelica Model One

2. График в OpenModelica уоказывет числоности армии X и Y. Армия Y побеждает.

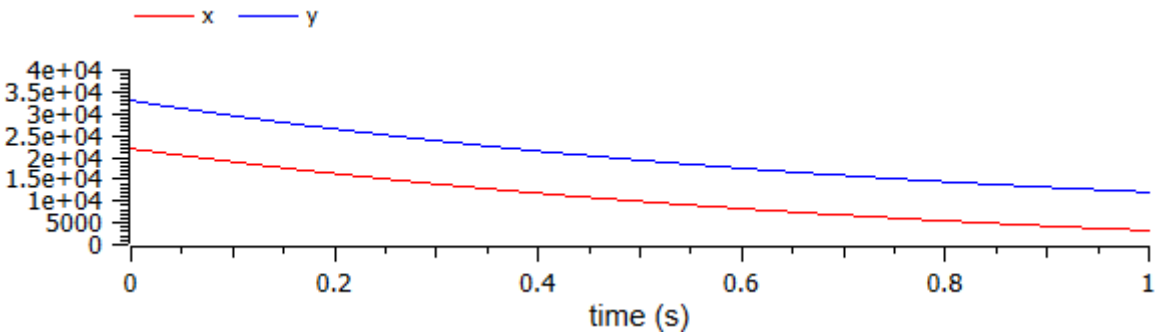

Рис. 8: OpenModelica Model One Graph

Модель ведение боевых действий с участием регулярных войск и партизанских отрядов

Модлеирование на языке программировании Julia

1. Все то же самое как для первой модели только величены разные и модель то же разнная.

```
In [4]: using Plots
        using DifferentialEquations

        x0 = 22022
        y0 = 33033

        a = 0.343
        b = 0.895
        c = 0.699
        h = 0.433

        P(t) = 2 * sin(2*t)
        Q(t) = 2 * cos(t)

        manpower = [x0, y0]
        p = (a, b, c, h)
        T = [0, 1]

        function F(du, u, p, t)
            a, b, c, h = p
            du[1] = -a * u[1] - b * u[2] + P(t)
            du[2] = -c * u[1]*u[2] - b * u[2] + Q(t)
        end

        prob = ODEProblem(F, manpower, T, p)

Out[4]: ODEProblem with uType Vector{Int64} and tType Int64. In-place: true
        timespan: (0, 1)
        u0: 2-element Vector{Int64}:
         22022
         33033
```

Рис. 9: Model Two Julia code part one

```
In [5]: sol = solve(prob)

Out[5]: retcode: Success
Interpolation: specialized 4th order "free" interpolation, specialized 2nd order "free" stiffness-aware interpolation
t: 86-element Vector{Float64}:
 0.0
 7.172847746826926e-5
 0.00010404765155495742
 0.0001573663865909362
 0.0002009957083640141
 0.00025199234322974945
 0.00030084947343353495
 0.0003524275056220109 4
 0.00040381304453938064
 0.00045633273866333595
 0.0005090711732929949
 0.000562355652377288
 0.0006158783701186638
 ⋮
 0.8171228592665443
 0.8370111972103351
 0.8561989520275517
 0.8747409264570462
 0.8926870763656339
 0.9100828710303033
 0.9269696556877294
 0.9433850076136675
 0.9593630753404611
 0.9749349001552963
 0.99012870690279
 1.0
```

Рис. 10: Model Two Julia code part two

2. График в Julia уоказывет числоности армии X и Y. Армия X побеждает.

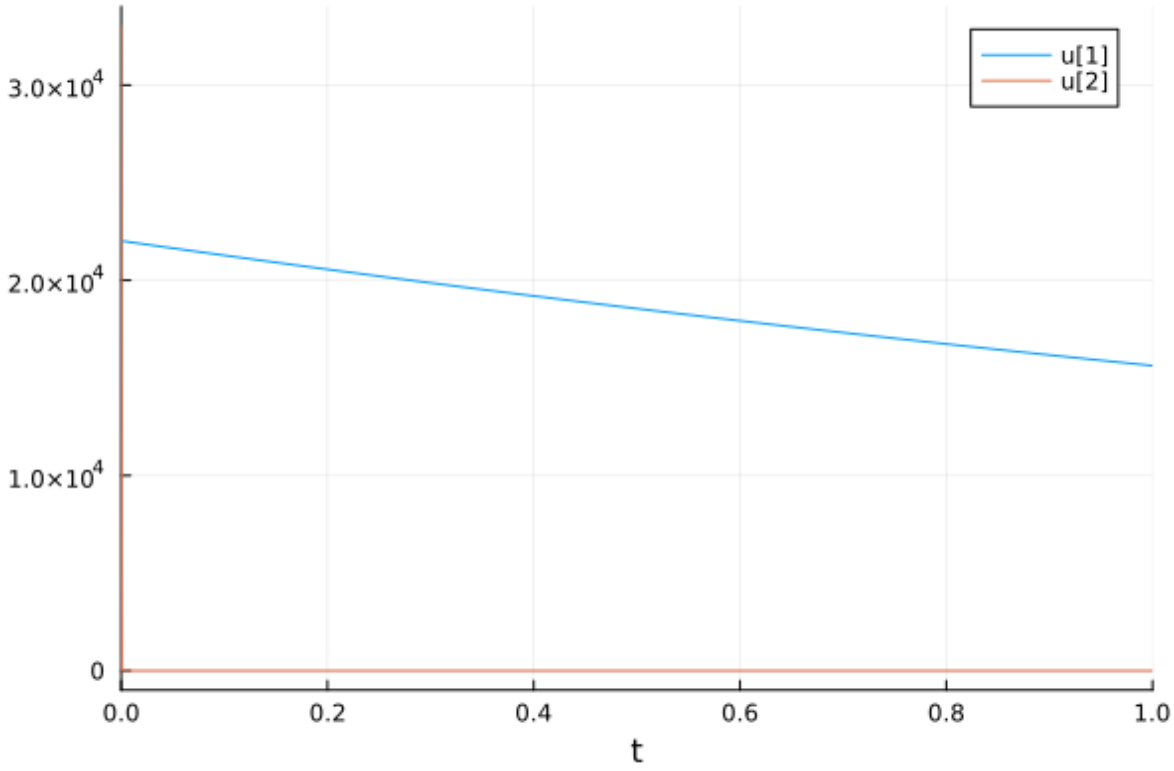

Рис. 11: Model Two Graph Julia

Модлеирование на языке программировании OpenModelica

1. Все то же самое как для первой модели только величены разные и модель то же разнная.

```
model lab3_2

Real x;
Real y;

Real a = 0.343;
Real b = 0.895;
Real c = 0.699;
Real h = 0.433;
Real t = time;

initial equation
x = 22022;
y = 33033;

equation
der(x) = -a*x - b*y + 2*sin(2*t);
der(y) = -c*x*y -b*y + 2*cos(t);

end lab3_2;
```

Рис. 12: Model Two OpenModelica

2. График в OpenModelica уоказывет числонности армии X и Y. Армия X побеждает.

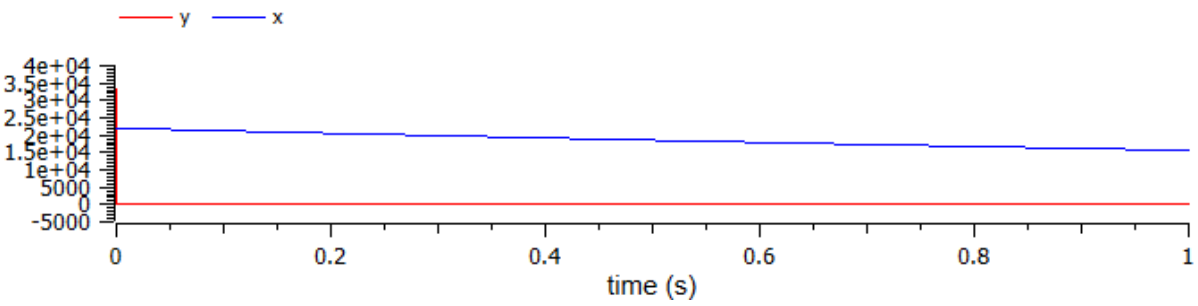

Рис. 13: Model Two Graph OpenModelica

Исходный код

Julia

1. Модель боевых действий между регулярными войсками

```
using Plots
using DifferentialEquations

x0 = 22022 #численность первой армии x
y0 = 33033 #численность второй армии y

a = 0.401 #константа, характеризующая степень влияния различных факторов на потер
b = 0.707 #эффективность боевых действий армии y
c = 0.606 #ффективность боевых действий армии x
h = 0.502 # константа, характеризующая степень влияния различных факторов на поте

P(t) = sin(8*t) #возможность подхода подкрепления к армии x
Q(t) = cos(6*t) #возможность подхода подкрепления к армии y

manpower = [x0, y0]
p = (a, b, c, h)
T = [0, 1] #итервал временни

function modelOne(du, u, p, t)
    a, b, c, h = p
    du[1] = -a * u[1] - b * u[2] + P(t)
    du[2] = -c * u[1] - b * u[2] + Q(t)
end

prob = ODEProblem(modelOne, manpower, T, p)
solOne = solve(prob)
plot(solOne)
```

2. Модель ведение боевых действий с участием регулярных войск и партизанских отрядов

```
using Plots
using DifferentialEquations

x0 = 22022
y0 = 33033

a = 0.343
b = 0.895
c = 0.699
h = 0.433

P(t) = 2 * sin(2*t)
Q(t) = 2 * cos(t)

manpower = [x0, y0]
p = (a, b, c, h)
T = [0, 1]

function F(du, u, p, t)
    a, b, c, h = p
    du[1] = -a * u[1] - b * u[2] + P(t)
    du[2] = -c * u[1]*u[2] - b * u[2] + Q(t)
end

prob = ODEProblem(F, manpower, T, p)
sol = solve(prob)
plot(sol)
```

OpenModelica

1. Модель боевых действий между регулярными войсками

```
model lab3_1

Real x;
Real y;

Real a = 0.401;
Real b = 0.707;
Real c = 0.606;
Real h = 0.502;
Real t = time;

initial equation
x = 22022;
y = 33033;

equation
der(x) = -a*x - b*y + sin(8*t);
der(y) = -c*x -b*y + cos(6*t);

end lab3_1;
```

2. Модель ведение боевых действий с участием регулярных войск и партизанских отрядов

```
model lab3_2

Real x;
Real y;
```

```
Real a = 0.343;
Real b = 0.895;
Real c = 0.699;
Real h = 0.433;
Real t = time;

initial equation
x = 22022;
y = 33033;

equation
der(x) = -a*x - b*y + 2*sin(2*t);
der(y) = -c*x*y -b*y + 2*cos(t);

end lab3_2;
```

Вывод

- В первом случае армия Y побеждает блягодря большее числонести армии. Во втором случае армия Y проеграла даже с большее количество солдатов из-за боейвих действей с пртизанами, а не с регулярной армию.
- В общем моделирвать математические процесии легче и быстрее в OpenModelica чем на Julia

Библиография

1. Julia 1.10 Documentation // Julia URL: https://docs.julialang.org/en/v1/ (дата обращения: 24.02.2024).
2. М. П. Осипов: к идентификации личности автора первой модели глобальных процессов. Дата обращения: 22 сентября 2020. Архивировано 29 сентября 2020 года. (из Wikipedia)