**Завдання**

1. Задати вихідні дані , , де  - номер студента в списку групи.
2. Запрограмувати рекурентні співвідношення - в системі Matlab (або Octava і т.і.).
3. Побудувати графік траєкторії судна
4. Дослідити залежність траєкторії от варіації параметрів , .
5. Розглянути задачу піймання цілі за умови, що ціль рухається зі швидкістю:
6. сталою за модулем
7. напрямком, що змінюється випадковим чином у заданих межах на кожній ітерації процесу.

**Реалізація моделі**

Модель була реалізована мовою C#. Посилання на git репозиторій: <https://github.com/Bohdan628318ylypchenko/Optimal-Control-Lab1.git>

Рекурентні співвідношення реалізовано за допомогою класу ShipNavigationProblem:

public static class ShipNavigationProblem

{

private static readonly int \_INIT\_LIST\_CAPACITY = 100;

private static readonly Random \_random = new Random();

public static TrajectoryInfo TrajectoryShip(Func<double, double> f, double s0, double v,

double l, double fi,

long N, long K, double epsilon)

{

List<V2> shipTrajectory = new(\_INIT\_LIST\_CAPACITY);

(double tau, double vtau, double vtau2) = \_InitializeTauVtauVtau2(l, v, N);

V2 destination = \_V2FromPolar(l, fi);

V2 p = new(0.0, 0.0);

V2 u;

shipTrajectory.Add(p);

for (long i = 0; i < N + K && !\_IsArrived(p, destination, epsilon); i++)

{

double langrandian =

Math.Sqrt(

Math.Pow(destination.x1 - p.x1 - s0 \* f(p.x2) \* tau, 2.0) +

Math.Pow(destination.x2 - p.x2, 2.0)

) \* vtau - vtau2;

u.x1 = ((destination.x1 - p.x1 - s0 \* f(p.x2) \* tau) \* vtau) / (langrandian + vtau2);

u.x2 = ((destination.x2 - p.x2) \* vtau) / (langrandian + vtau2);

p.x1 = p.x1 + (s0 \* f(p.x2) + v \* u.x1) \* tau;

p.x2 = p.x2 + u.x2 \* vtau;

shipTrajectory.Add(p);

}

return new TrajectoryInfo(shipTrajectory, new List<V2> { destination, destination },

tau, shipTrajectory.Count \* tau);

}

public static TrajectoryInfo TrajectoryShipAndDestination(Func<double, double> f, double s0, double vShip,

double l, double fi,

long N, long K, double epsilon,

double vDestination, double aMin, double aMax)

{

List<V2> shipTrajectory = new(\_INIT\_LIST\_CAPACITY);

List<V2> destinationTrajectory = new(\_INIT\_LIST\_CAPACITY);

(double tau, double vtau, double vtau2) = \_InitializeTauVtauVtau2(l, vShip, N);

double a;

double aRange = aMax - aMin;

V2 destination = \_V2FromPolar(l, fi);

V2 p = new(0.0, 0.0);

V2 u;

shipTrajectory.Add(p);

destinationTrajectory.Add(destination);

for (long i = 0; i < N + K && !\_IsArrived(p, destination, epsilon); i++)

{

a = aMin + \_random.NextDouble() \* aRange;

destination.x1 = destination.x1 + (s0 \* f(destination.x2) + vDestination \* Math.Cos(a)) \* tau;

destination.x2 = destination.x2 + vDestination \* Math.Sin(a) \* tau;

double langrandian =

Math.Sqrt(

Math.Pow(destination.x1 - p.x1 - s0 \* f(p.x2) \* tau, 2.0) +

Math.Pow(destination.x2 - p.x2, 2.0)

) \* vtau - vtau2;

u.x1 = ((destination.x1 - p.x1 - s0 \* f(p.x2) \* tau) \* vtau) / (langrandian + vtau2);

u.x2 = ((destination.x2 - p.x2) \* vtau) / (langrandian + vtau2);

p.x1 = p.x1 + (s0 \* f(p.x2) + vShip \* u.x1) \* tau;

p.x2 = p.x2 + u.x2 \* vtau;

shipTrajectory.Add(p);

destinationTrajectory.Add(destination);

}

return new TrajectoryInfo(shipTrajectory, destinationTrajectory,

tau, shipTrajectory.Count \* tau);

}

private static (double, double, double) \_InitializeTauVtauVtau2(double l, double v, double N)

{

double tau = l / (v \* N);

double vtau = v \* tau;

double vtau2 = vtau \* vtau;

return (tau, vtau, vtau2);

}

private static V2 \_V2FromPolar(double l, double fi)

{

return new V2(l \* Math.Cos(fi), l \* Math.Sin(fi));

}

private static bool \_IsArrived(V2 a, V2 b, double epsilon)

{

return Math.Sqrt(Math.Pow(b.x1 - a.x1, 2.0) + Math.Pow(b.x2 - a.x2, 2.0)) <= epsilon;

}

}

**Дослідження траєкторії (нерухома ціль)**

Номер студента у списку – 15.

Початкові параметри обрано згідно умови задачі:

Маємо:

A screenshot of a computer

Description automatically generated(N – кількість розбиттів часу ; epsilon – параметр, що використовується у додатковій умові зупинки: «якщо відстань між ціллю і кораблем менша рівна epsilon – вважати ціль досягнутою.)

A graph with lines and numbers

Description automatically generated

A screenshot of a computer program

Description automatically generated



Вхідні / вихідні параметри дослідження 1

**Очікувано, корабель не досягнув цілі: зростаюча функція f, однакові початкові швидкості корабля та течії.**

Змінимо функцію f на спадну:

A screenshot of a computer

Description automatically generated

Маємо:

A graph of a function

Description automatically generated

A screenshot of a computer program

Description automatically generated



Як видно з графіку, вхідних / вихідних даних, **корабель досяг цілі.** Вплив течії помітно лише на перших ітераціях: функція зміни є агресивно спадною.

Замінимо функцію f на логарифмічну, збільшимо швидкість човна:

A screenshot of a computer

Description automatically generated

Маємо:

A graph with a curved line

Description automatically generated

A screenshot of a computer program

Description automatically generated

Ціль не досягнута. Очевидно, вплив функції f є більшим за зміну сталої швидкості v.

Проведемо останнє дослідження з нерухомою ціллю: вкотре змінимо вид f, додатково збільшимо v.

Маємо:

A screenshot of a computer

Description automatically generated

A graph with a curve

Description automatically generated

A screenshot of a computer program

Description automatically generated

Ціль досягнуто, хоча траєкторія є дещо неочікуваною….

**Дослідження траєкторії (рухома ціль)**

Рух цілі задається трьома параметрами:

* – модуль швидкості цілі
* - межі кута руху цілі. Остаточне значення кута генерується випадковим чином для кожної ітерації (кути задаються в радіанах).

Крім власного руху, на ціль також впливає течія.

Проведемо перше дослідження з наступними параметрами:

A screenshot of a computer

Description automatically generated

A graph with lines and dots

Description automatically generated

A screenshot of a computer program

Description automatically generated

Течія очікувано знесла ціль, не зважаючи на те, що напрямок власного руху цілі є протилежним. Корабель досяг цілі.

Спробуємо підібрати такий модуль швидкості цілі, щоб ціль зміщувалась вліво, долаючи опір течії.

A screenshot of a computer

Description automatically generated

A graph with lines and a line

Description automatically generated

A screenshot of a computer program

Description automatically generated

З графіка бачимо, що деякий час ціль рухалась проти течії. При цьому течія потроху зносила ціль вгору, щоразу збільшуючи значення функції f. Тому в деякий момент течія знову почала зносити ціль праворуч.

Збільшимо швидкість човна, щоб досягнути цілі:

A screenshot of a computer

Description automatically generated

A graph with lines and dots

Description automatically generated

A screenshot of a computer program

Description automatically generated

Збільшення швидкості човна дозволило швидко наздогнати ціль. Зменшимо швидкість човна, щоб отримати більш цікаву траєкторію:

A screenshot of a computer program

Description automatically generated

A graph of a graph

Description automatically generated

A screenshot of a computer program

Description automatically generated