Завдання

1. Задати вихідні дані $s_0 = \sqrt{n}, v = \sqrt{n}, l = n, \varphi = n\pi/25$, $f(x_2) = x_2$, де n - номер студента в списку групи.
2. Запрограмувати рекурентні співвідношення **Error! Reference source not found.**-**Error! Reference source not found.** в системі Matlab (або Octava і т.і.).
3. Побудувати графік траєкторії судна $\{(x_1(t_k), x_2(t_k)),\ t_k = \tau k,\ k = 0,1,...,N,N+1,...,K\}$
4. Дослідити залежність траєкторії от варіації параметрів s_0, v, l, φ, N.
5. Розглянути задачу піймання цілі за умови, що ціль рухається зі швидкістю:
    a. сталою за модулем
    b. напрямком, що змінюється випадковим чином у заданих межах $\alpha_{min}, \alpha_{max}$ 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:

$s_0 = \sqrt{15} = 3.8729;$

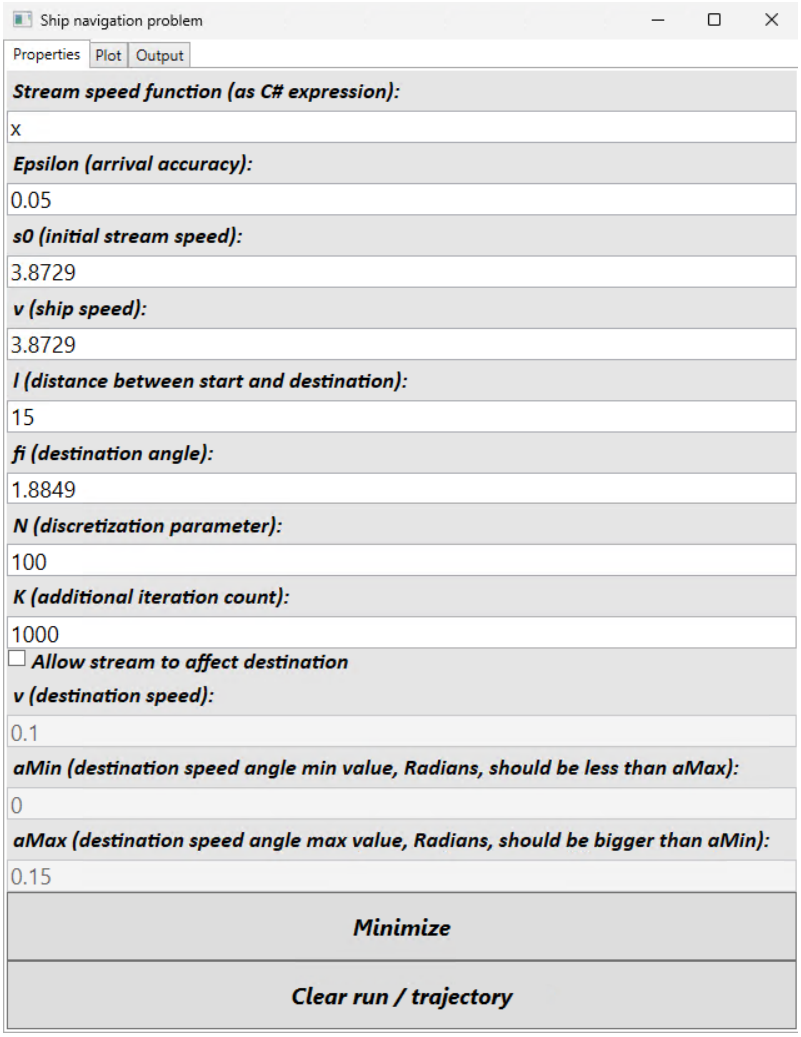
$v_{ship} = \sqrt{15} = 3.8729;$

$l = 15$

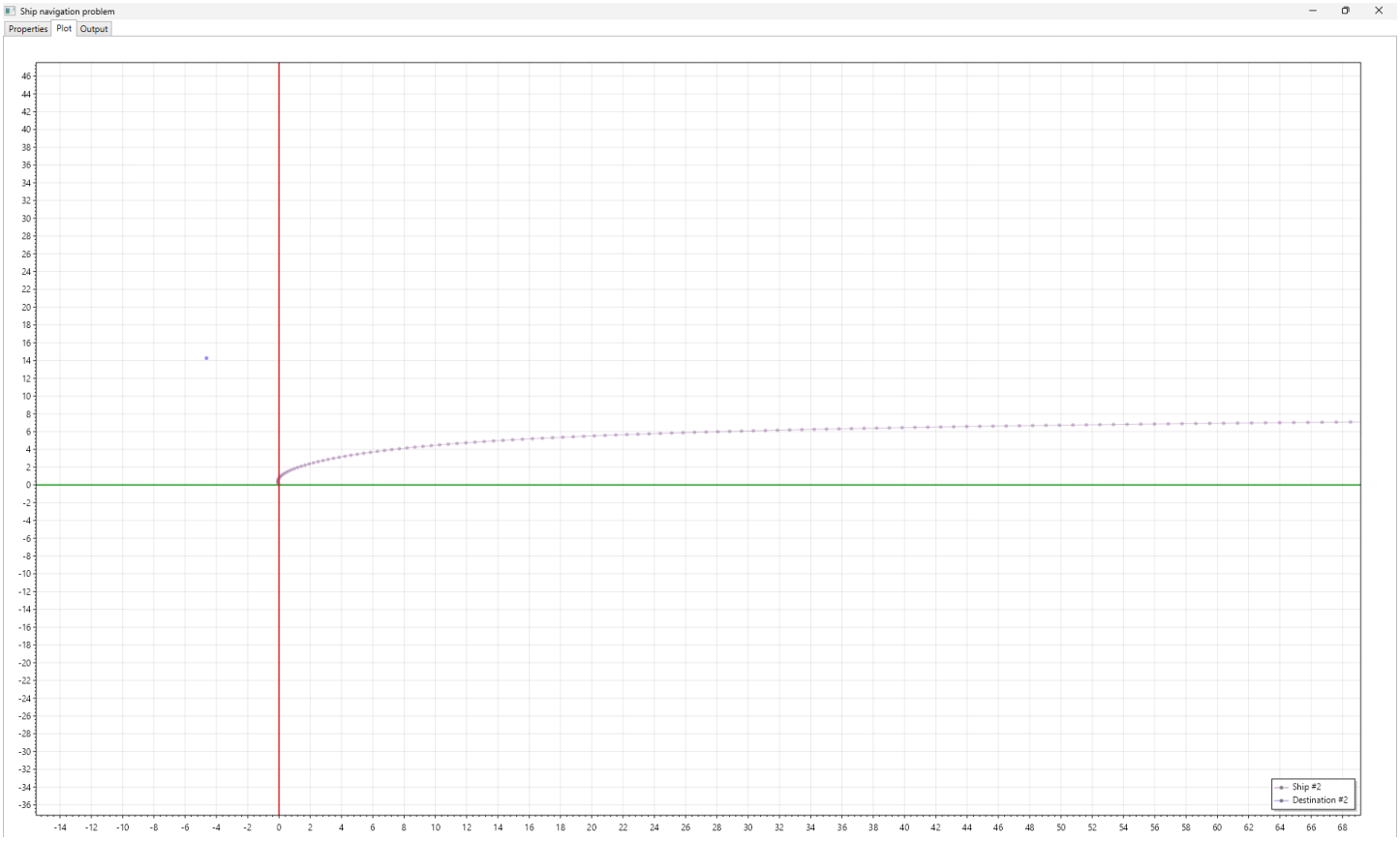
$\varphi = \frac{15\pi}{25} = \frac{3\pi}{5} = 1.8849$

$f(x_2) = x_2$

Маємо:



(N – кількість розбиттів часу $t = \frac{l}{v}$; epsilon – параметр, що використовується у додатковій умові зупинки: «якщо відстань між ціллю і кораблем менша рівна epsilon – вважати ціль досягнутою.)

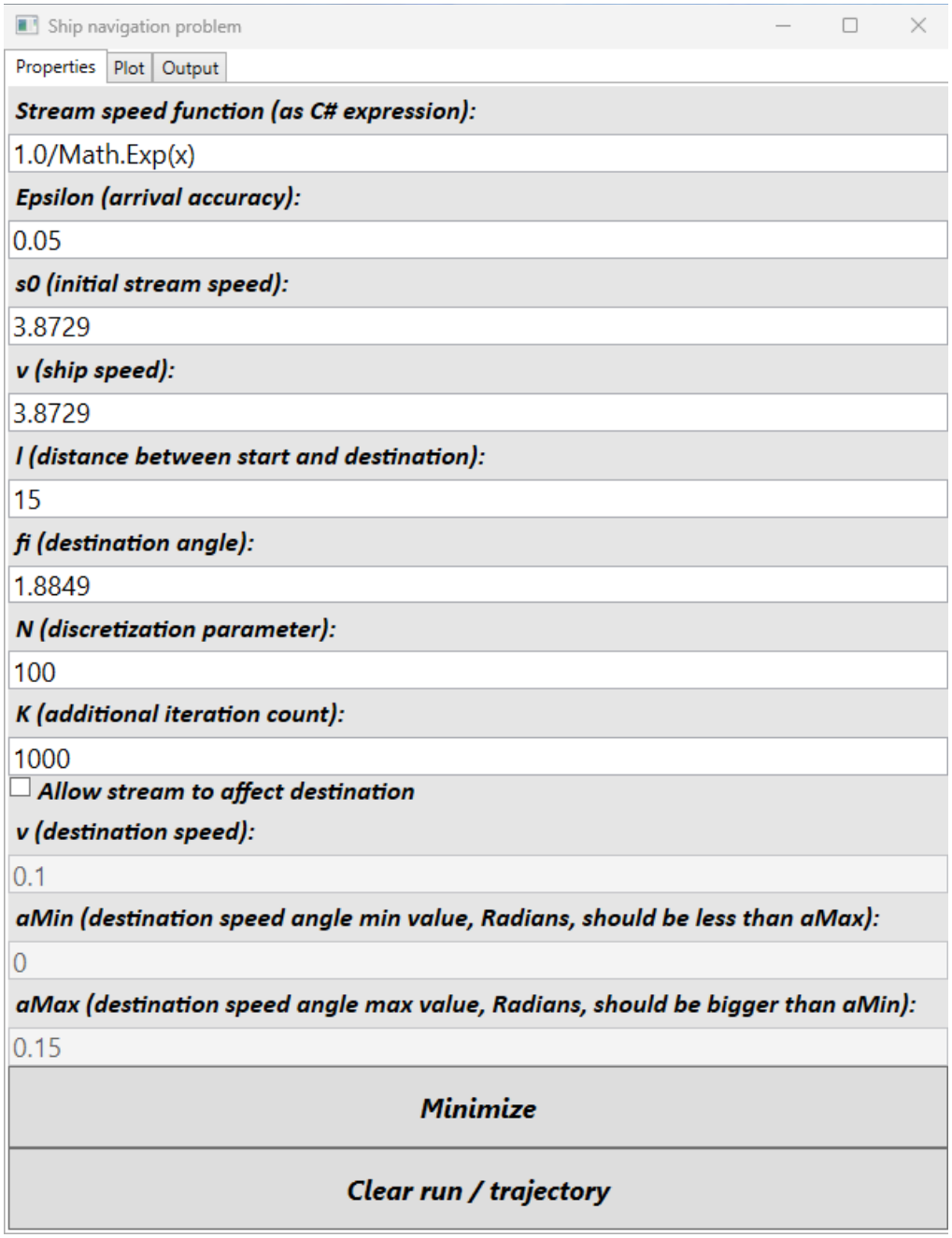



```
═══════════⇒ Run 2 ═══════════
input parameters:
    f       = x
    s0      = 3.8729
    vShip   = 3.8729
    l       = 15
    fi      = 1.8849
    epsilon = 0.05
    N       = 100
    K       = 1000
    vDest   = 0.1
    aMin    = 0
    aMax    = 0.15
trajectory:
    ship trajectory start        = (0;0)
    ship trajectory end          = (1221.575580328729;9.376341406241444)
    destination trajectory start = (-4.634461839259042;14.266105406187483)
    destination trajectory end   = (-4.634461839259042;14.266105406187483)
    tau                          = 0.03873066694208474
    total time                   = 42.6424643032353
═══════════════════════════════════════
```

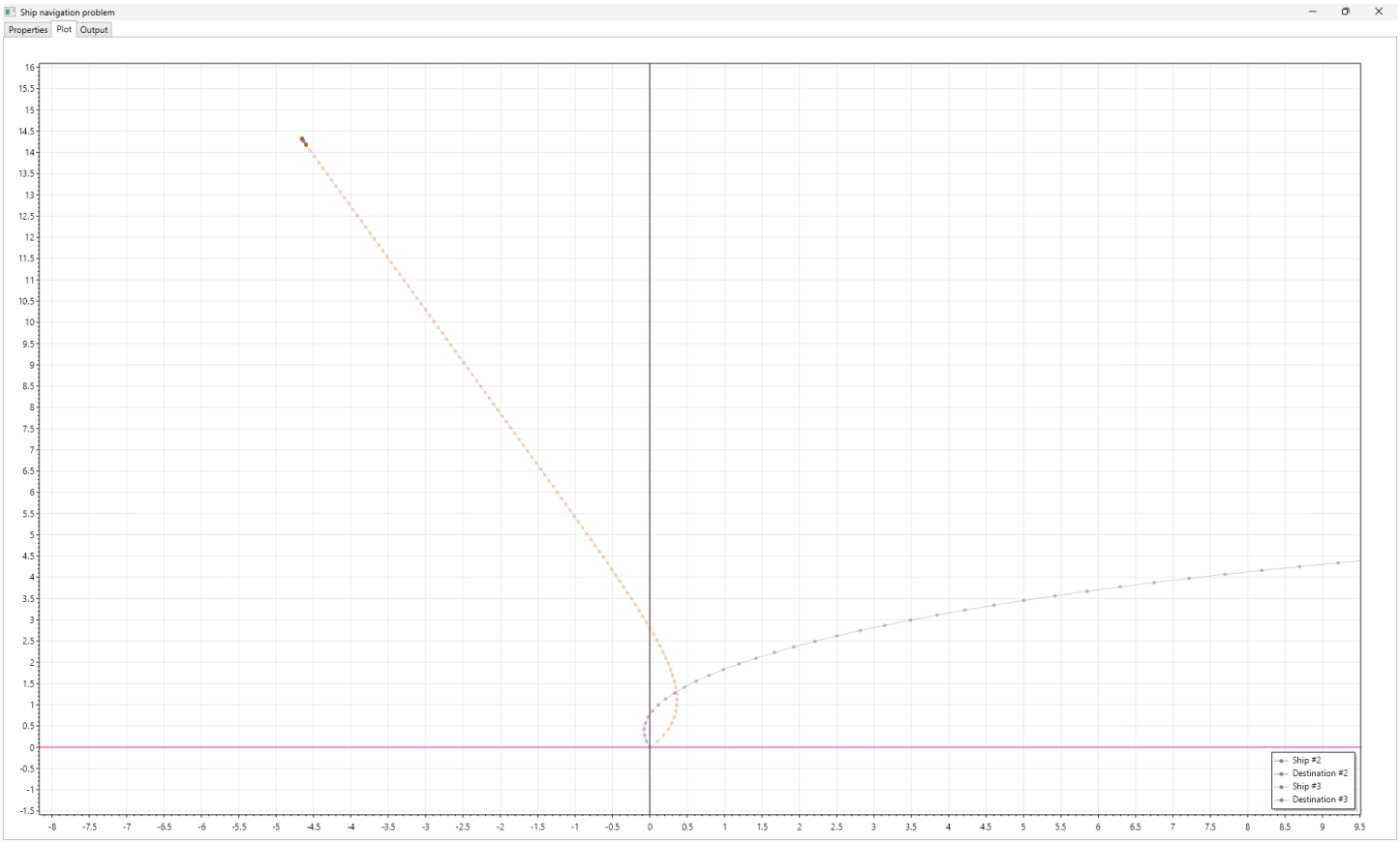
Вхідні / вихідні параметри дослідження 1

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

Змінимо функцію f на спадну:


Ship navigation problem
Properties
Plot
Output
Stream speed function (as C# expression):
1.0/Math.Exp(x)
Epsilon (arrival accuracy):
0.05
s0 (initial stream speed):
3.8729
v (ship speed):
3.8729
l (distance between start and destination):
15
fi (destination angle):
1.8849
N (discretization parameter):
100
K (additional iteration count):
1000
Allow stream to affect destination
v (destination speed):
0.1
aMin (destination speed angle min value, Radians, should be less than aMax):
0
aMax (destination speed angle max value, Radians, should be bigger than aMin):
0.15
Minimize
Clear run / trajectory

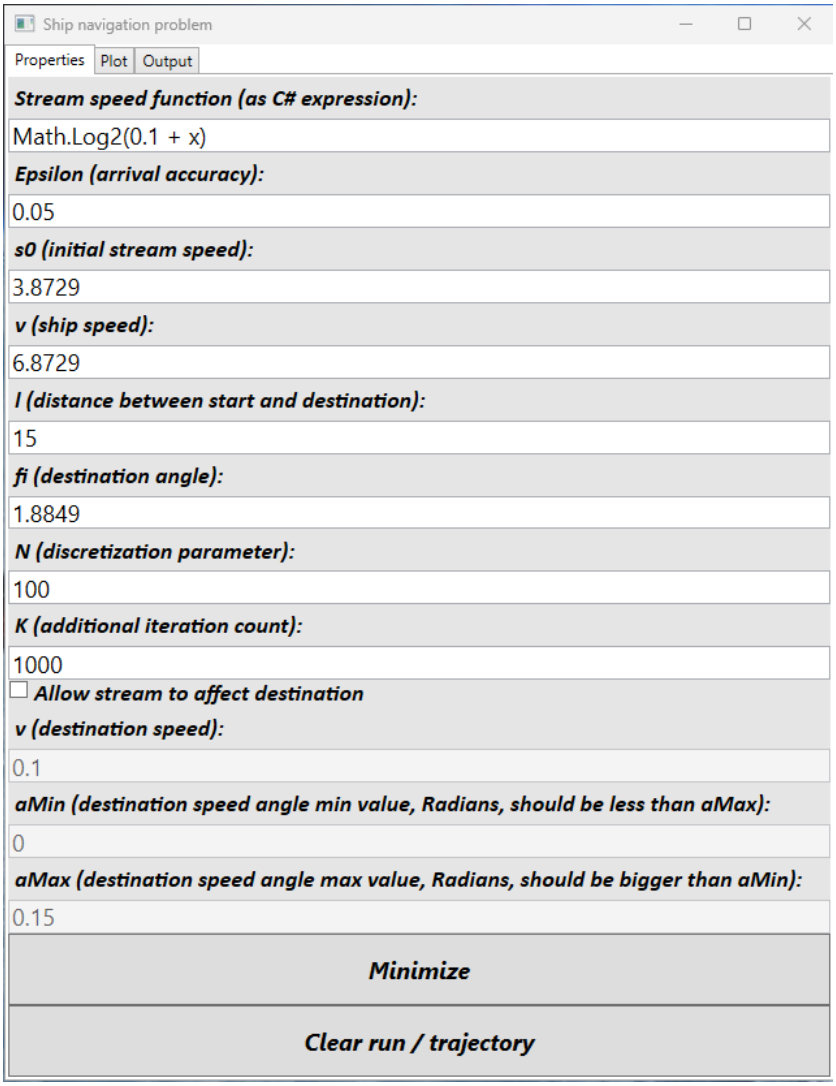
Маємо:



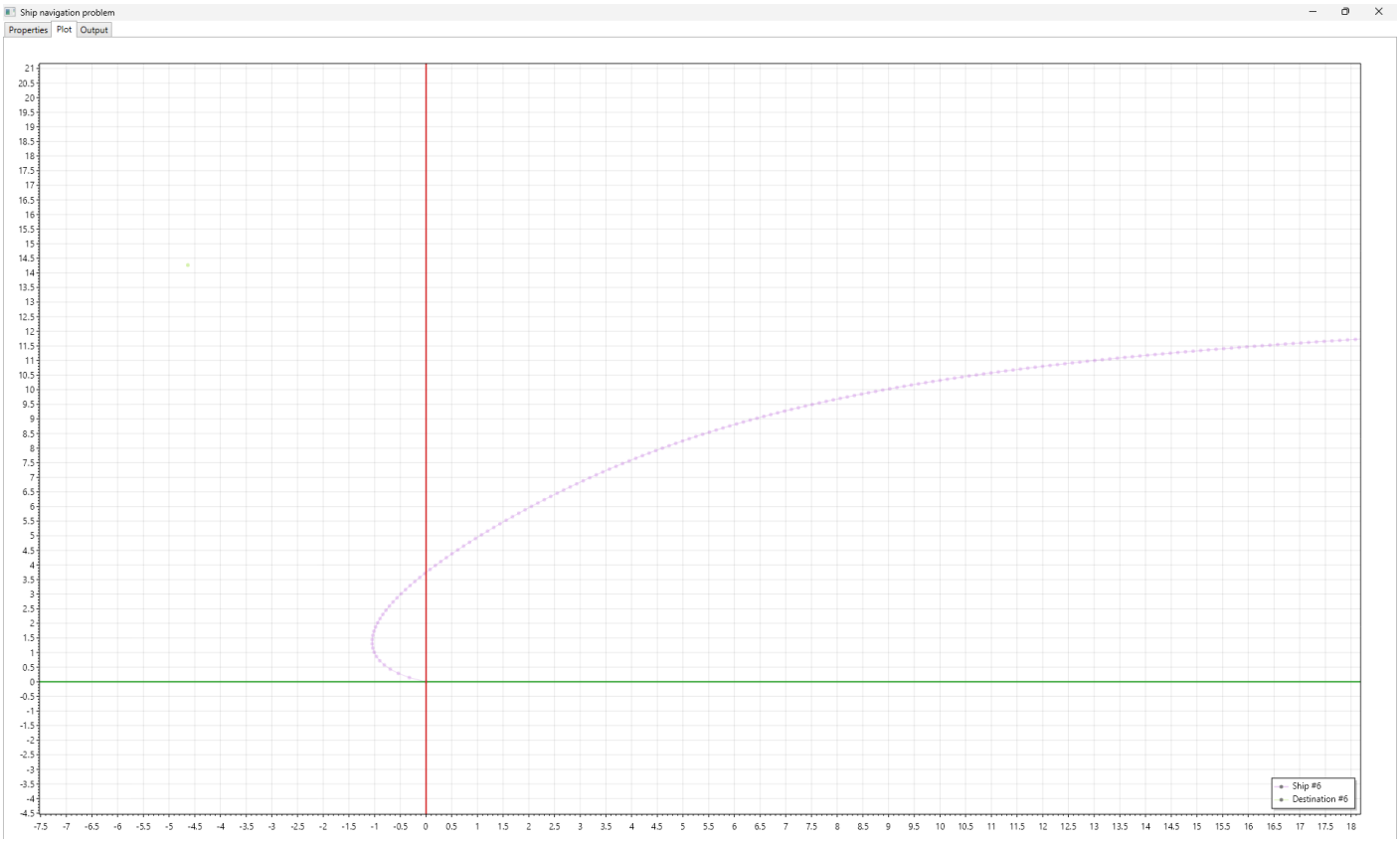

```
═══════════⇒ Run 3 ═══════════
input parameters:
    f       = 1.0/Math.Exp(x)
    s0      = 3.8729
    vShip   = 3.8729
    l       = 15
    fi      = 1.8849
    epsilon = 0.05
    N       = 100
    K       = 1000
    vDest   = 0.1
    aMin    = 0
    aMax    = 0.15
trajectory:
    ship trajectory start        = (0;0)
    ship trajectory end          = (-4.599189446403512;14.180275607759084)
    destination trajectory start = (-4.634461839259042;14.266105406187483)
    destination trajectory end   = (-4.634461839259042;14.266105406187483)
    tau                          = 0.03873066694208474
    total time                   = 42.6424643032353
═══════════════════════════════════════
```

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:


Ship navigation problem
Properties
Plot
Output
Stream speed function (as C# expression):
Math.Log2(0.1 + x)
Epsilon (arrival accuracy):
0.05
s0 (initial stream speed):
3.8729
v (ship speed):
6.8729
l (distance between start and destination):
15
fi (destination angle):
1.8849
N (discretization parameter):
100
K (additional iteration count):
1000
Allow stream to affect destination
v (destination speed):
0.1
aMin (destination speed angle min value, Radians, should be less than aMax):
0
aMax (destination speed angle max value, Radians, should be bigger than aMin):
0.15
Minimize
Clear run / trajectory

Маємо:

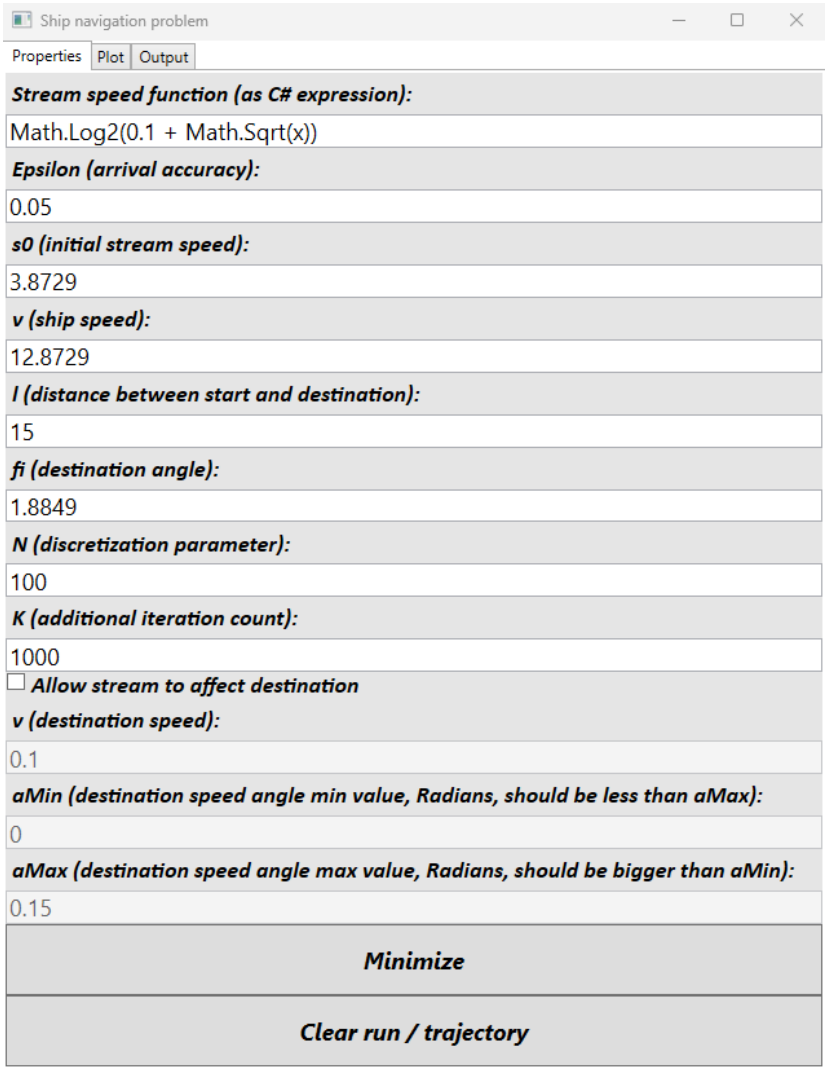


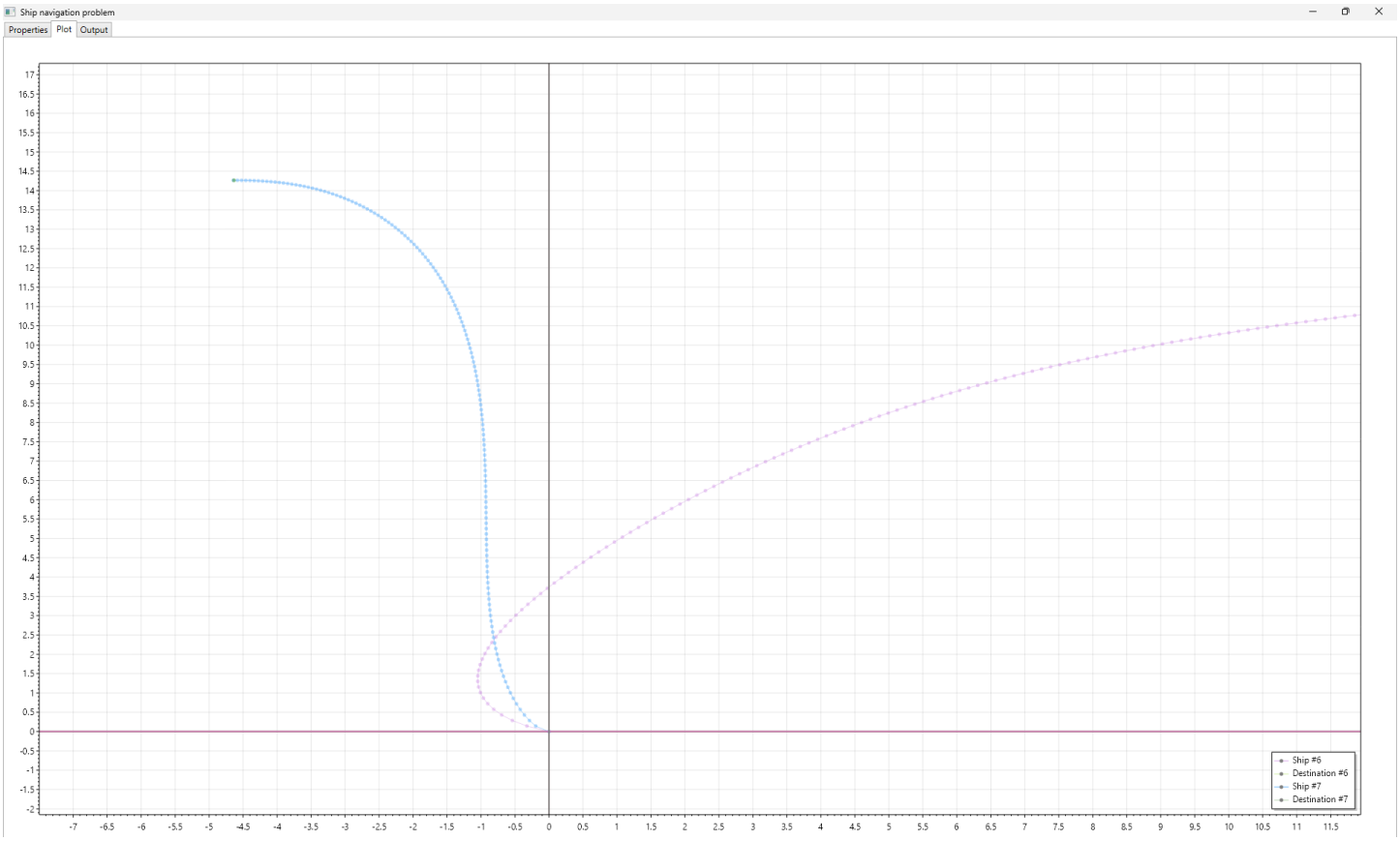

```
═══════════⟹ Run 6 ═══════════
input parameters:
    f       = Math.Log2(0.1 + x)
    s0      = 3.8729
    vShip   = 6.8729
    l       = 15
    fi      = 1.8849
    epsilon = 0.05
    N       = 100
    K       = 1000
    vDest   = 0.1
    aMin    = 0
    aMax    = 0.15
trajectory:
    ship trajectory start        = (0;0)
    ship trajectory end          = (175.8024547302296;13.8778001404104 68)
    destination trajectory start = (-4.634461839259042;14.266105406187483)
    destination trajectory end   = (-4.634461839259042;14.266105406187483)
    tau                          = 0.021824848317304194
    total time                   = 24.029157997351916
═══════════════════════════════════════
```

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:


Ship navigation problem
Properties
Plot
Output
Stream speed function (as C# expression):
Math.Log2(0.1 + Math.Sqrt(x))
Epsilon (arrival accuracy):
0.05
s0 (initial stream speed):
3.8729
v (ship speed):
12.8729
l (distance between start and destination):
15
fi (destination angle):
1.8849
N (discretization parameter):
100
K (additional iteration count):
1000
Allow stream to affect destination
v (destination speed):
0.1
aMin (destination speed angle min value, Radians, should be less than aMax):
0
aMax (destination speed angle max value, Radians, should be bigger than aMin):
0.15
Minimize
Clear run / trajectory




```
═══════════⟹ Run 7 ═══════════
input parameters:
    f       = Math.Log2(0.1 + Math.Sqrt(x))
    s0      = 3.8729
    vShip   = 12.8729
    l       = 15
    fi      = 1.8849
    epsilon = 0.05
    N       = 100
    K       = 1000
    vDest   = 0.1
    aMin    = 0
    aMax    = 0.15
trajectory:
    ship trajectory start        = (0;0)
    ship trajectory end          = (-4.6454716101936 41;14.266132073028977)
    destination trajectory start = (-4.634461839259042;14.266105406187483)
    destination trajectory end   = (-4.634461839259042;14.266105406187483)
    tau                          = 0.011652386020244 08
    total time                   = 1.7245531309961237
═══════════════════════════════════════
```

Ціль досягнуто, хоча траєкторія є дещо неочікуваною....

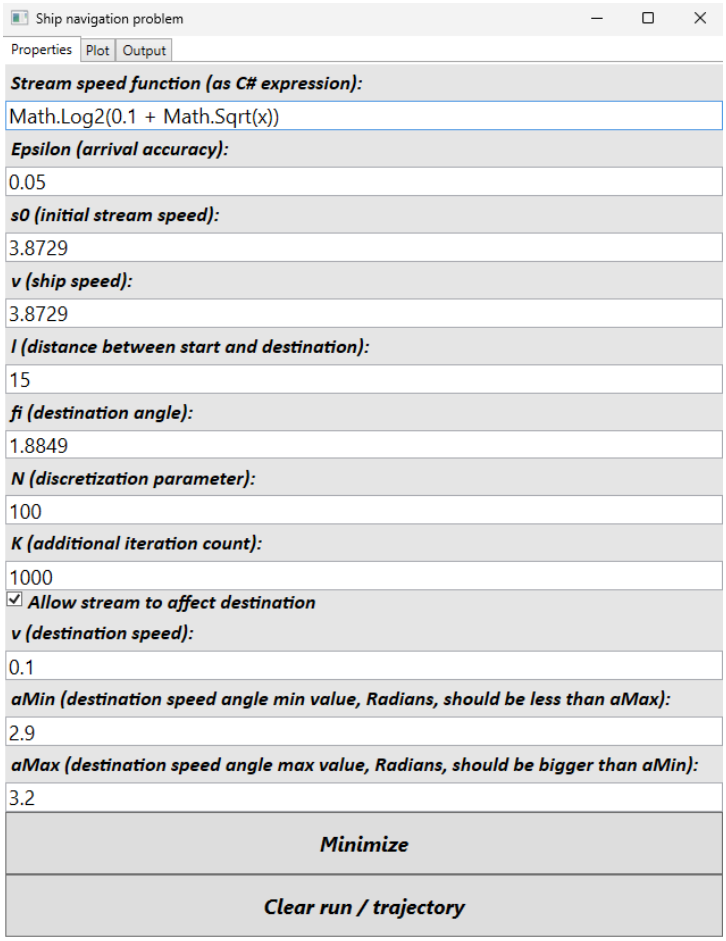
Дослідження траєкторії (рухома ціль)

Рух цілі задається трьома параметрами:

- v_{dest} – модуль швидкості цілі
- $\alpha_{min}; \alpha_{max}$- 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:




```
═══════════⟹ Run 12 ═══════════
input parameters:
    f       = Math.Log2(0.1 + Math.Sqrt(x))
    s0      = 3.8729
    vShip   = 3.8729
    l       = 15
    fi      = 1.8849
    epsilon = 0.05
    N       = 100
    K       = 1000
    vDest   = 0.1
    aMin    = 2.9
    aMax    = 3.2
trajectory:
    ship trajectory start        = (0;0)
    ship trajectory end          = (26.033910400056705;14.2740446802427 76)
    destination trajectory start = (-4.634461839259042;14.266105406187483)
    destination trajectory end   = (26.056113612054585;14.302701584223424)
    tau                          = 0.038730666942084 74
    total time                   = 4.144181362803067
═══════════════════════════════════════
```

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.

Ship navigation problem

Properties | Plot | Output

Stream speed function (as C# expression):

Math.Log2(0.1 + Math.Sqrt(x))

Epsilon (arrival accuracy):

0.05

s0 (initial stream speed):

3.8729

v (ship speed):

3.8729

l (distance between start and destination):

15

fi (destination angle):

1.8849

N (discretization parameter):

100

K (additional iteration count):

1000

☑ ***Allow stream to affect destination***

v (destination speed):

10

aMin (destination speed angle min value, Radians, should be less than aMax):

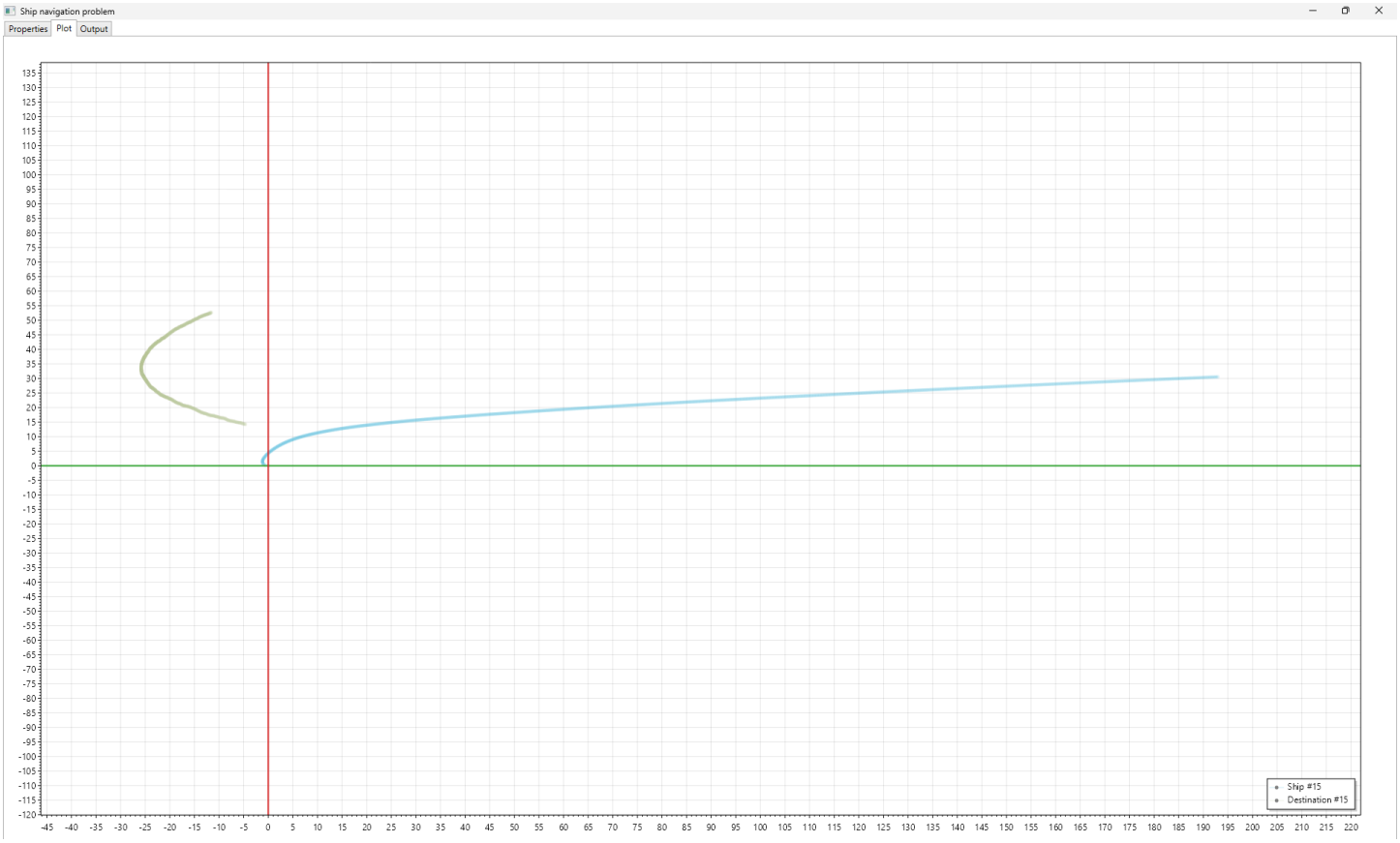
2.9

aMax (destination speed angle max value, Radians, should be bigger than aMin):

3.2

Minimize

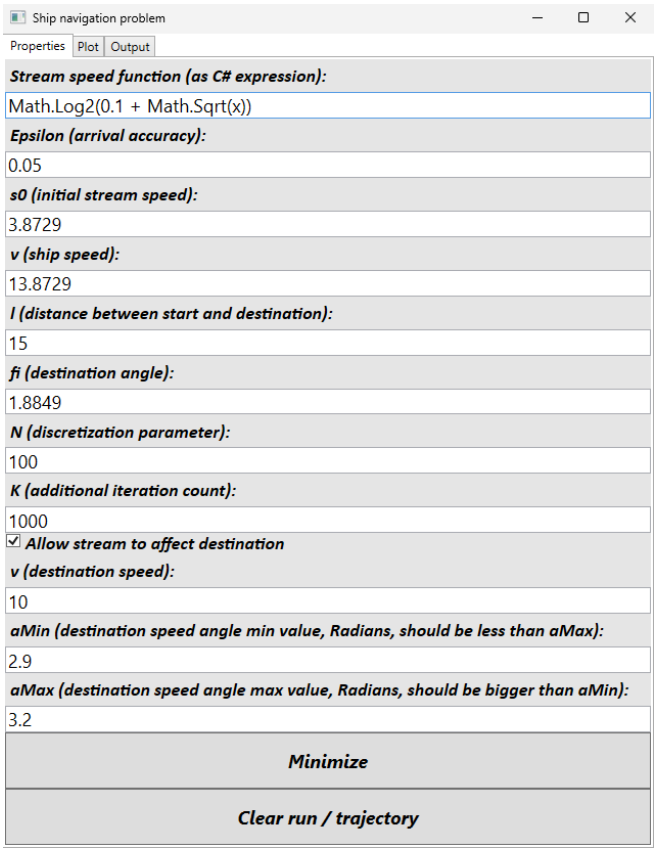
Clear run / trajectory

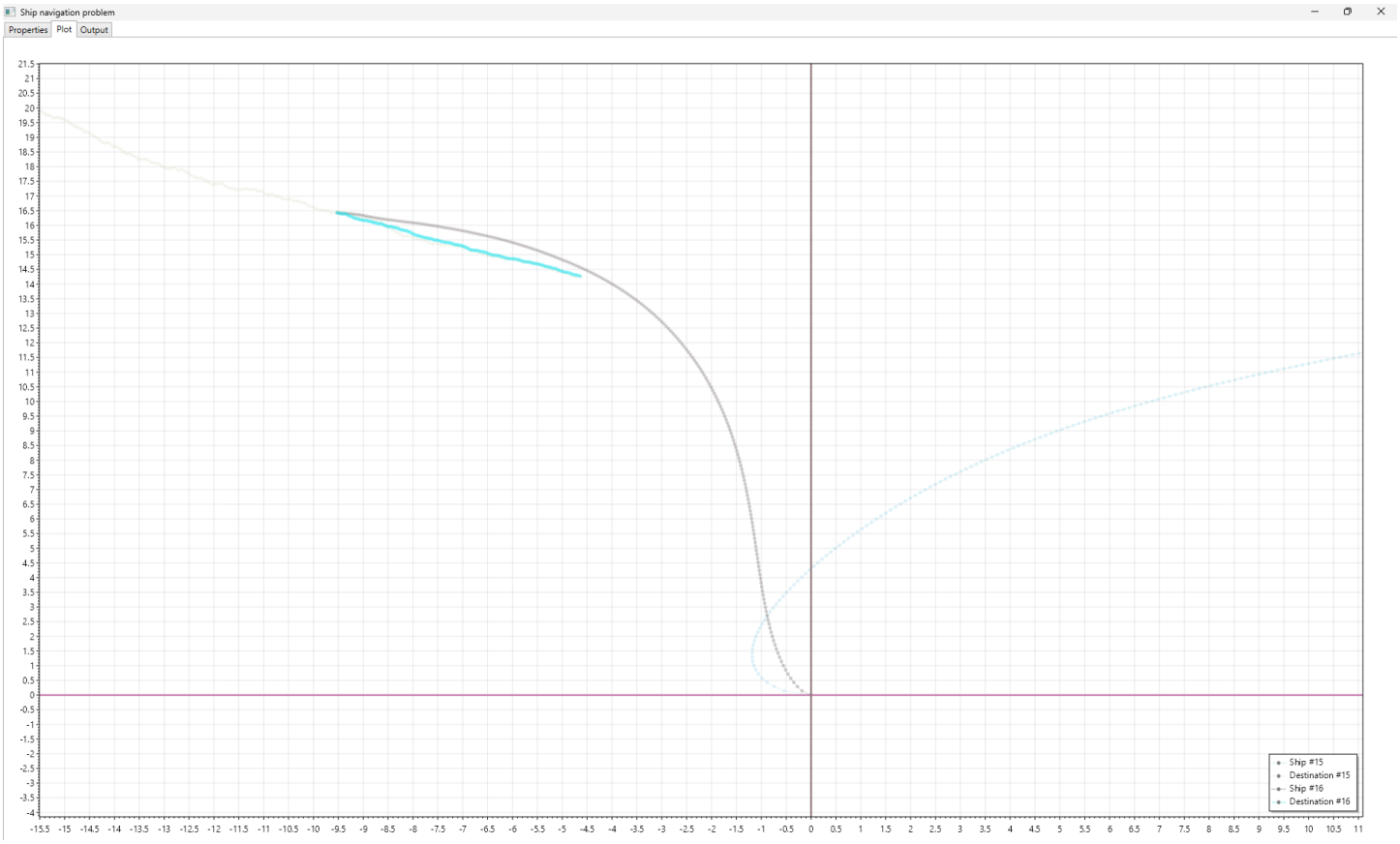


```
═══════════⟹ Run 15 ═══════════════
input parameters:
    f       = Math.Log2(0.1 + Math.Sqrt(x))
    s0      = 3.8729
    vShip   = 3.8729
    l       = 15
    fi      = 1.8849
    epsilon = 0.05
    N       = 100
    K       = 1000
    vDest   = 10
    aMin    = 2.9
    aMax    = 3.2
trajectory:
    ship trajectory start        = (0;0)
    ship trajectory end          = (192.94694508749862;30.542008580994356)
    destination trajectory start = (-4.634461839259042;14.266105406187483)
    destination trajectory end   = (-11.578747817883725;52.62999349913917)
    tau                          = 0.03873066694208474
    total time                   = 42.6424643032353
═══════════════════════════════════════
```

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:






```
═══════════⟹ Run 16 ═══════════════
input parameters:
    f       = Math.Log2(0.1 + Math.Sqrt(x))
    s0      = 3.8729
    vShip   = 13.8729
    l       = 15
    fi      = 1.8849
    epsilon = 0.05
    N       = 100
    K       = 1000
    vDest   = 10
    aMin    = 2.9
    aMax    = 3.2
trajectory:
    ship trajectory start        = (0;0)
    ship trajectory end          = (-9.51840035510263;16.430966151933166)
    destination trajectory start = (-4.634461839259042;14.266105406187483)
    destination trajectory end   = (-9.529305265052052;16.432373600712264)
    tau                          = 0.010812447289319465
    total time                   = 2.2922388253357266
═══════════════════════════════════════
```

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

Ship navigation problem

Properties | Plot | Output

Stream speed function (as C# expression):

Math.Log2(0.1 + Math.Sqrt(x))

Epsilon (arrival accuracy):

0.05

s0 (initial stream speed):

3.8729

v (ship speed):

10.2

l (distance between start and destination):

15

fi (destination angle):

1.8849

N (discretization parameter):

100

K (additional iteration count):

2000

☑ ***Allow stream to affect destination***

v (destination speed):

10

aMin (destination speed angle min value, Radians, should be less than aMax):

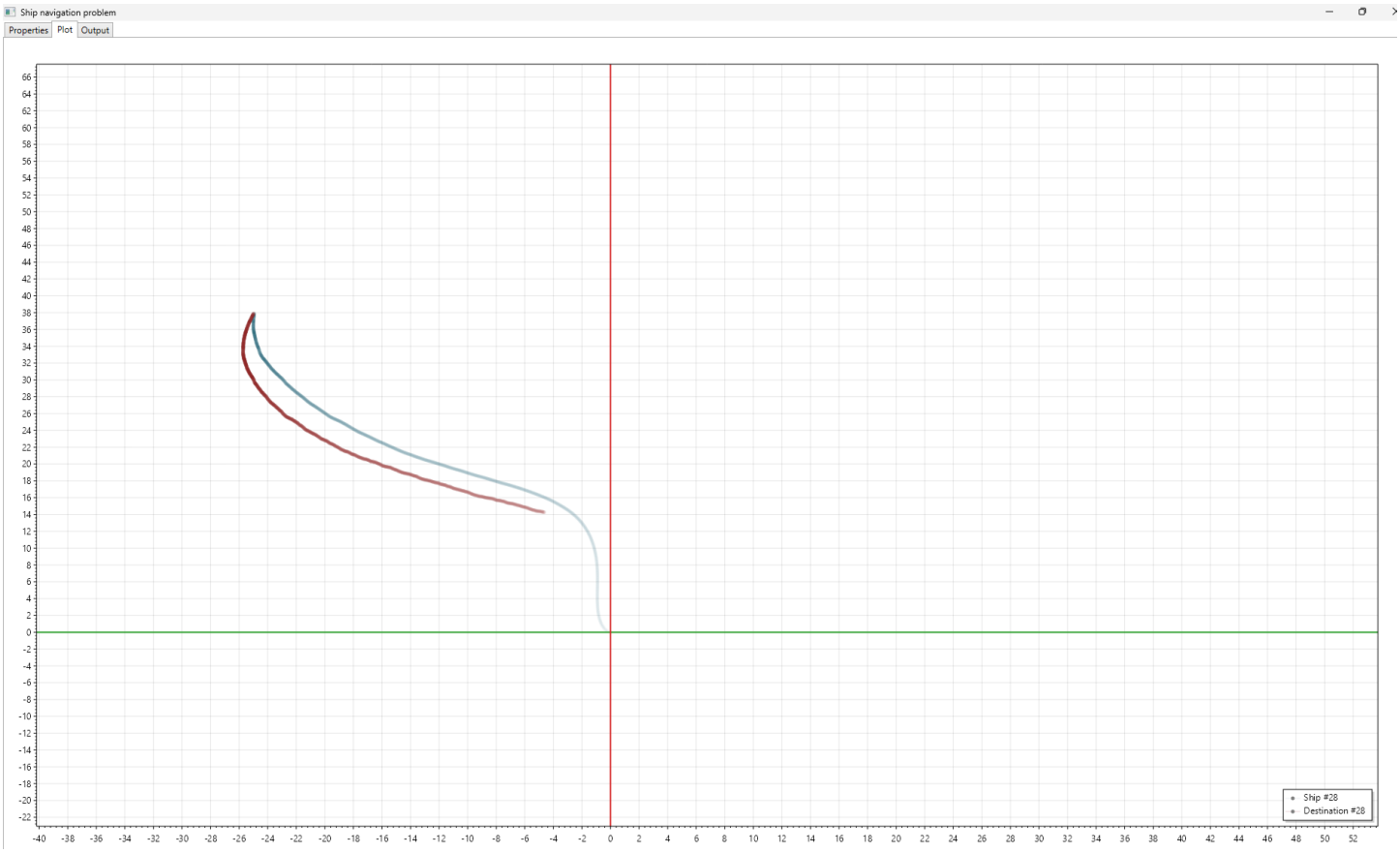
2.9

aMax (destination speed angle max value, Radians, should be bigger than aMin):

3.2

Minimize

Clear run / trajectory



```
═══════⟹ Run 28 ═══════════════
input parameters:
    f       = Math.Log2(0.1 + Math.Sqrt(x))
    s0      = 3.8729
    vShip   = 10.2
    l       = 15
    fi      = 1.8849
    epsilon = 0.05
    N       = 100
    K       = 2000
    vDest   = 10
    aMin    = 2.9
    aMax    = 3.2
trajectory:
    ship trajectory start        = (0;0)
    ship trajectory end          = (-24.93764224633785;37.90273905429983)
    destination trajectory start = (-4.634461839259042;14.266105406187483)
    destination trajectory end   = (-24.98690780403325;37.90876497720281)
    tau                          = 0.0147058823529411 78
    total time                   = 25.632352941176475
═══════════════════════════════════════════
```