EINDHOVEN UNIVERSITY OF TECHNOLOGY

Control Principles for Engineered Systems

Problem set IV solutions: Networked Control System

1 Algebraic graph theory: Incidence matrix, Laplacian, connectivity and rigidity

Consider the following (unweighted) undirected graph in Fig. 1.

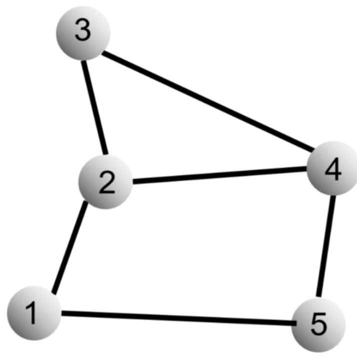

Figure 1: An undirected graph

1. Give the incidence matrix and the Laplacian matrix for the graph in Fig. 1.

   By following the row-edge column-vertex convention and specifying an arbitrary orientation in each edge, one can define an incidence matrix as below

$$H = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 1 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{1}$$

   The Laplacian matrix is given by

$$L = H^T H = \begin{bmatrix} 2 & -1 & 0 & 0 & -1 \\ -1 & 3 & -1 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & -1 & 3 & -1 \\ -1 & 0 & 0 & -1 & 2 \end{bmatrix} \tag{2}$$

   Note: With different orientations in each edge and different index for all edges, the expression of the incidence matrix H may look different. However, the Laplacian matrix shall be the same.

   Note: one can also use the definition $L = D - A$ to construct the Laplacian matrix. You will get the same Laplacian matrix.

2. Give the spectrum (all eigenvalues) of the Laplacian matrix (via e.g., Matlab).

   By Matlab function eig(L), we can obtain the spectrum of the Laplacian matrix

$$\lambda(L) = [0, 1.3820, 2.3820, 3.6180, 4.6180] \tag{3}$$

3. Show that the graph is not rigid. Can you add one edge to the graph to make it rigid?

   The graph is not rigid. Note that for a 2-D framework with n agents to be rigid, it should contain at least $2n-3$ edges. The graph has 5 vertices and 6 edges (fewer than the minimum number $2 \times 5 - 3 = 7$ edges) and is not rigid.

   One can construct the rigidity matrix R, choose a generic configuration position p, and evaluate the rank $R(p)$ to determine the rigidity of the graph. It is easy to show that for the framework in the graph (under generic positions), $\text{rank}(R(p)) = 6 < 7$, therefore the graph is not rigid.

   One can add one edge $(1,4)$ or $(2,5)$ to make the graph rigid. The rank of the rigidity matrix for the new graph (with new edge $(1,4)$ or $(2,5)$) is 7, which meets the rigidity condition.

4. (Open research question): connectivity management for mobile networks is an active research topic in multi-agent systems. A naive idea to maintain connectivity for mobile networks is to preserve all existing edges in a connected graph when node dynamics are moving (i.e., when one edge exists, it should be preserved for all time). There are many other options for connectivity management in a dynamic environment.
   In the lecture we have discussed the algebraic connectivity (the second smallest eigenvalue of the graph Laplacian) as an index to quantify the connectivity of a static/dynamic graph. A possible solution is to maintain the positivity of this eigenvalue for a mobile graph. Read the survey paper [1] and summarize some key possible solutions on connectivity control of mobile robot networks.

   This question aims to evaluate students' ability of writing a concise technical review on research papers. You shall review (in your own words) the following approaches on connectivity management control:

   - Optimization-based connectivity control (centralized and distributed connectivity maximization approaches)
   - Continuous feedback connectivity control
   - Hybrid feedback connectivity control

2 Robotic coordination: continuous-time cyclic pursuit on the plane

(Modified from Exercise 1.6 of [2])

Consider four mobile robots on a plane with positions $p_i \in \mathbb{R}^2, i \in \{1,2,3,4\}$, and moving according to $\dot{p}_i = u_i$, where $u_i \in \mathbb{R}^2$ are the velocity commands. The task of the robots is rendezvous at a common point (while using only onboard sensors). A simple strategy to achieve rendezvous is cyclic pursuit: each robot i picks another robot, say $i+1$, and pursues it. (Here we follow the convention $4+1 \to 1$.) In other words, we set $u_i = p_{i+1} - p_i$ and obtain the closed-loop system (see also corresponding simulation below in Fig. 2):

$$\begin{bmatrix}\dot{p}_1\\ \dot{p}_2\\ \dot{p}_3\\ \dot{p}_4\end{bmatrix} = \left(\begin{bmatrix}-1 & 1 & 0 & 0\\ 0 & -1 & 1 & 0\\ 0 & 0 & -1 & 1\\ 1 & 0 & 0 & -1\end{bmatrix} \otimes I_2\right)\begin{bmatrix}p_1\\ p_2\\ p_3\\ p_4\end{bmatrix} \tag{4}$$

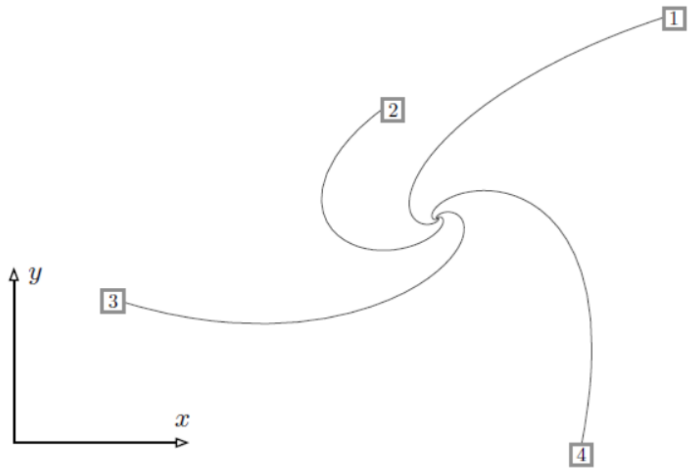

Figure 2: Robot rendezvous via cyclic pursuit.

1. Find the eigenvalues of the underlying Laplacian matrix.

   The Laplacian matrix is

$$L = \begin{bmatrix}1 & -1 & 0 & 0\\ 0 & 1 & -1 & 0\\ 0 & 0 & 1 & -1\\ -1 & 0 & 0 & 1\end{bmatrix} \tag{5}$$

   and its eigenvalues are

$$\lambda(L) = [0, 1+i, 1-i, 2] \tag{6}$$

   Note: the matrix in (4) is not Laplacian (noting the different sign).

   Note: the matrix with the Kronecker product is not Laplacian.

2. Show that the average robot position $average(p(t)) = \sum_{i=1}^{4} p_i(t)/4$ remains constant for all $t \geq 0$;

   To show that the average robot position $\bar{p}(t) = average(p(t)) = \sum_{i=1}^{4} p_i(t)/4$ remains constant, we show that

$$\dot{\bar{p}}(t) = \sum_{i=1}^{4} \dot{p}_i(t)/4 = 0 \tag{7}$$

which is true since

$$\dot{p}_1 + \dot{p}_2 + \dot{p}_3 + \dot{p}_4 = 0 \tag{8}$$

Alternatively, the Laplacian matrix L has a left null vector $q_1 = [1, 1, 1, 1]^T$, which implies that the average point is invariant.

3. Prove that the robots asymptotically rendezvous at the initial average robot position mass, that is,

$$\lim_{t\to\infty} p_i(t) = average(p(0)), \quad i \in \{1, 2, 3, 4\} \tag{9}$$

The underlying graph is directed and has a rooted out-branching; therefore by the consensus dynamics (4) all robots converge to a consensus point (i.e., rendezvous). Since the average position is invariant, the converged rendezvous point must be the average position of initial points.

Alternatively, since the underlying directed graph has a rooted out-branching and is balanced, the consensus dynamics (4) ensures all robots' positions converge to an average consensus point (i.e., average rendezvous).

You can also prove this statement by using the matrix decomposition approach in the lecture slides.

Note: be careful in using the Lyapunov approach to prove the convergence. The Lyapunov function for *undirected* graph that we discussed in the lecture does not apply for *directed* graph in this question.

4. Show that if the robots are initially arranged in a square formation, then they remain in a square formation (the scaling of the shape is changing).

There are multiple ways to prove this statement.

Idea 1: show that the inner product of each adjacent relative position is zero: $(p_{i-1}(t) - p_i(t))^T(p_i(t) - p_{i+1}(t)) = 0$, and the inner product of the relative position in the cross edge is zero: $(p_i(t) - p_{i+2}(t))^T(p_{i-1}(t) - p_{i+1}(t)) = 0$, $\forall t \geq 0$.

Idea 2: show that the distances of the four edges are the same: $\|p_1(t) - p_2(t)\| = \|p_2(t) - p_3(t)\| = \|p_3(t) - p_4(t)\| = \|p_4(t) - p_1(t)\|$, $\forall t \geq 0$.

Idea 3: solve the linear system (4) with initial positions forming a square, and show that each edge distance is the same.

5. Simulate the four-robot dynamics of the system (1) under different initial conditions to verify the conclusions (2-4). Plot the trajectories of the four robots. You will expect to have similar trajectories as shown in Fig. 2.

Simulation examples:

6. (Open question) Can you speed up the convergence rate of the rendezvous by smartly adding one edge in the underlying directed graph? Justify your answer.

Adding any new edge to the underlying cyclic directed graph will increase the convergence rate of the rendezvous. This can be explained by the change of spectrum of the Laplacian. For example, if we add a new directed edge (1,3)

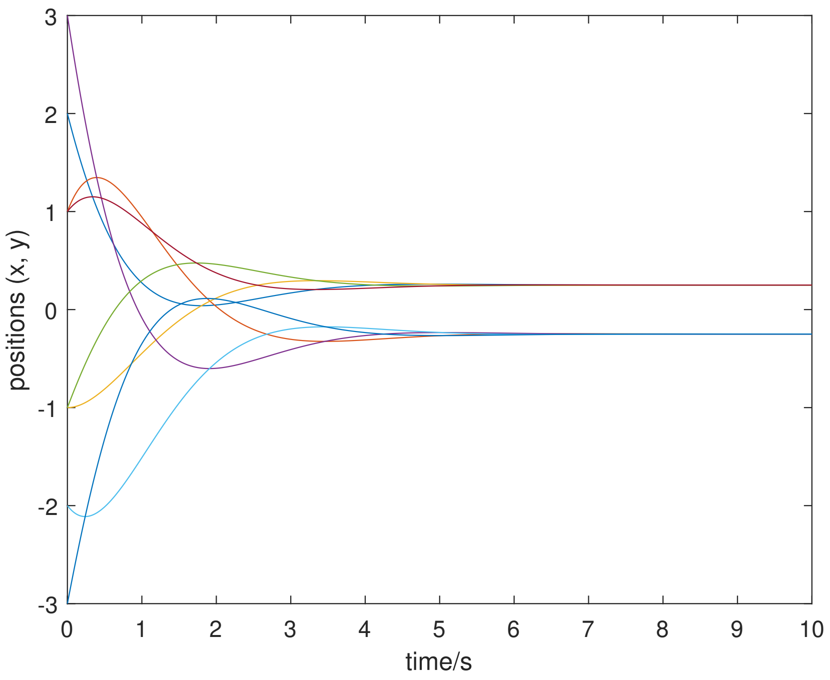

Figure 3: Rendezvous convergence of all robots' trajectories (in x, y coordinates).

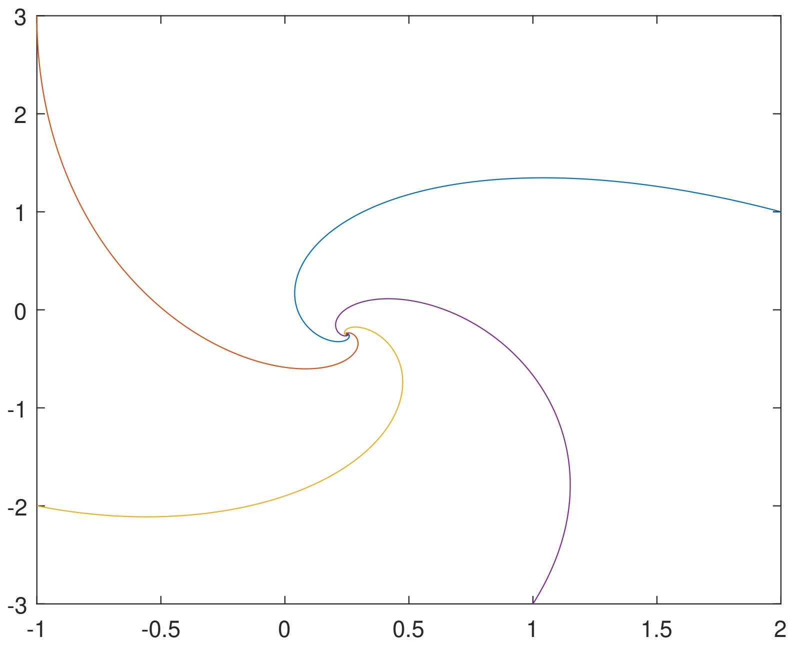

Figure 4: Trajectories of four robots in cyclic pursuit.

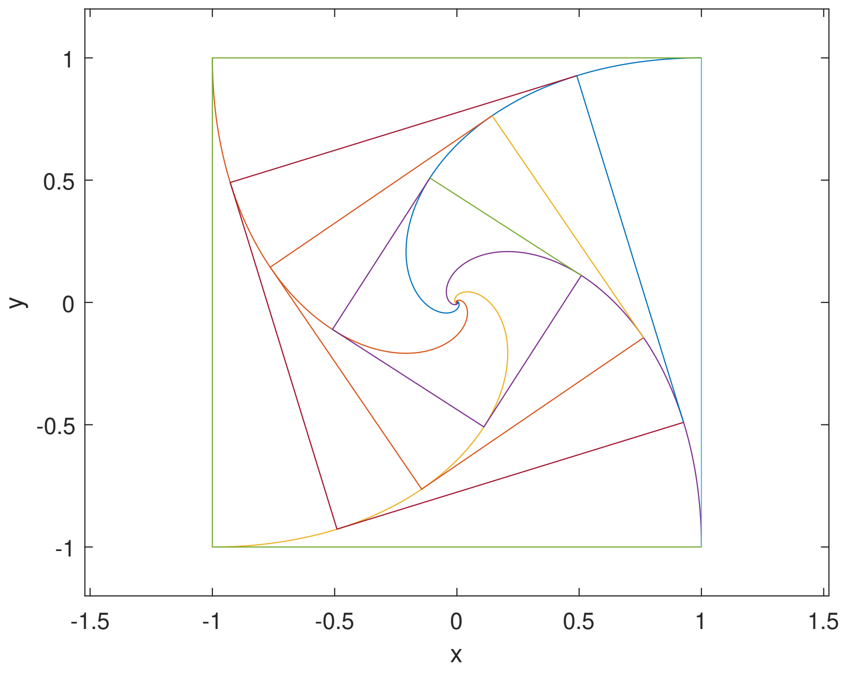

Figure 5: Trajectories of four robots in cyclic pursuit maintaining a square formation (with initial positions that form a square)

in the graph (i.e., robot 1 chases both robot 2 and robot 3), then the Laplacian matrix L for the new graph is

$$L = \begin{bmatrix} 2 & -1 & -1 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

and its eigenvalues are

$$\lambda(L) = [0, 1.5 + 0.866i, 1.5 - 0.866i, 2] \tag{11}$$

The real part of the second eigenvalue has increased, and the convergence rate is increased.

A simulation comparison with the new graph (a new directed edge (1,3)) is shown in Fig. 6.

Note: with the new directed edge (1,3) the directed graph is not balanced, therefore average consensus (rendezvous to the average point) is no longer available.

3 Formation control with obstacle avoidance

(Modified from Exercise 6.5 of [3])

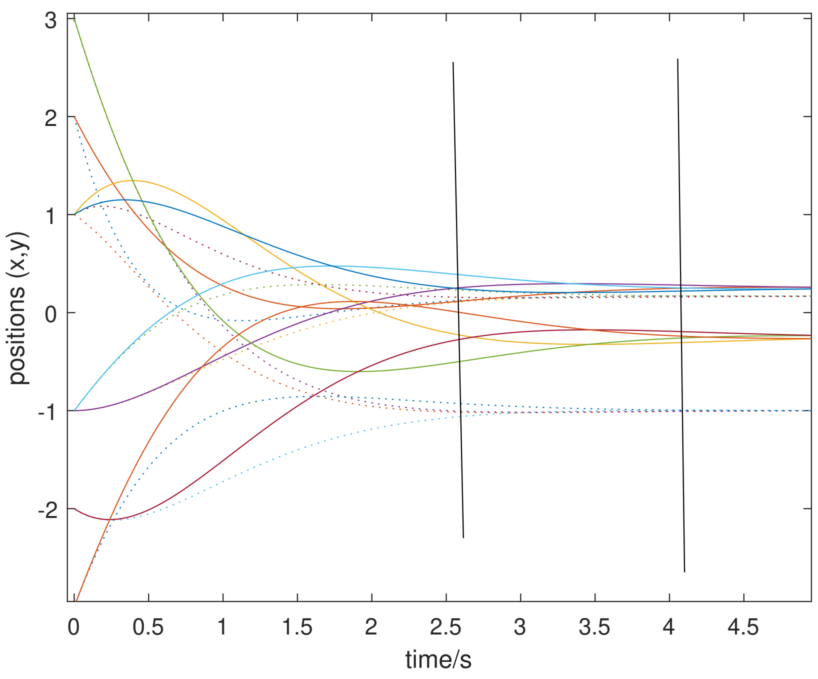

Figure 6: Convergence of robots' trajectories to rendezvous. The solid lines indicate the trajectories with the original graph, and the dashes lines indicate the trajectories with the new graph with an added edge $(1, 3)$

If a team of robots is to drive in formation while avoiding obstacles as well as progressing toward a goal location, one can, for example, let the individual agent dynamics be given by

$$\dot{p}_i = F_{form} + F_{goal} + F_{obst}, \tag{12}$$

where F_{form} is used to maintain formations. However, F_{goal} is used to steer the robot towards a goal and F_{obst} is used to have it avoid obstacles.

1. Find reasonable F_{goal} and F_{obst} (if you are unable to design F_{obst}, search in Google Scholar via keywords such as "collision avoidance" "obstacle avoidance" and you will find lots of research papers that will give you some hints).

2. Simulate your proposed solution (you can use either displacement-based or distance-based approach for F_{form}.). The final result should look something like the plots in Fig. 3.

This is an open research problem and there is no fixed answer to it. The following are examples of feasible controllers:

- Formation controller examples:

  - Displacement-based control (z^*_{ij} denotes the desired relative position between robots i and j):

$$F_{form} = \sum_{j \in N_i} (p_j - p_i) - z^*_{ij}) \tag{13}$$

  - Distance-based control (d_{ij} denotes the desired distance between robots i and j):

$$F_{form} = \sum_{j \in N_i} (\|p_i - p_j\|^2 - d_{ij}^2)(p_j - p_i) \tag{14}$$

- Goal controller examples: (p_{goal} denotes the position of the goal point)
  - Steering robot i to the goal point

$$F_{goal} = p_{goal} - p_i \tag{15}$$

    and let other robots to follow robot i (via the formation controller).
  - Steering all robots to the goal point

$$F_{goal,i} = p_{goal} - p_i, \forall i \tag{16}$$

  - Steering the centroid position of all robots to the goal point

$$F_{goal,i} = p_{goal} - \bar{p}, \tag{17}$$

    where $\bar{p} = \sum_{i=1}^{N} p_i / N$ is the group centroid.
  - Steering robot i to reach a desired distance $d*$ to the goal point

$$F_{goal,i} = (\|p_i - p_{goal}\|^2 - d*^2)(p_{goal} - p_i) \tag{18}$$

  - A more practical solution is to impose the goal controller F_{goal} to one robot, while all other robots are followers that are steered to the goal point
- Obstacle avoidance controller examples:

  F_{obst} is a function of distance-to-obstacle, and should provide an expelling force when a robot is too close to an obstacle (i.e., F_{obst} should drive a robot away when it approaches to an obstacle.). The following are candidates of obstacle avoidance controllers:

  - Consider an obstacle j with position q_j.

$$F_{obst,ij} = \frac{p_i - q_j}{\|p_i - q_j\|} \tag{19}$$

    One can also replace the denominator function by other functions, such as $F_{obst,ij} = \frac{p_i - q_j}{\|p_i - q_j\|^2}$. The control $F_{obst,ij}$ avoids collision of robot i to obstacle j.
  - Consider $F_{obst,ij}$ as a function of distance-to-obstacle $\|p_i - q_j\|$, and is only activated when the distance-to-obstacle reaches a threshold; the following is an example

$$F_{obst,ij} = \frac{p_i - q_j}{\|p_i - q_j\|}, \text{ if } \|p_i - q_j\| \leq \bar{R}, \text{ and } F_{obst,ij} = 0, \text{ if } \|p_i - q_j\| \geq \bar{R}, \tag{20}$$

    where $\bar{R}$ is a user-specified distance threshold.

- Consider an obstacle j with position q_j. Define the obstacle circle region with a radius r_j: $Q_j = \{\bar{q}_j : \|\bar{q}_j - q_j\| \leq r_j\}$ such that any robot cannot reach or enter Q_j. Then the obstacle avoidance controller F_{obst} should be designed such that the pointing direction is outside of the cone (shown in Fig. 7). With the coordination law all robots can reach a formation towards a goal point while avoid all circular obstacle regions (an illustration is shown in Fig. 8).

-

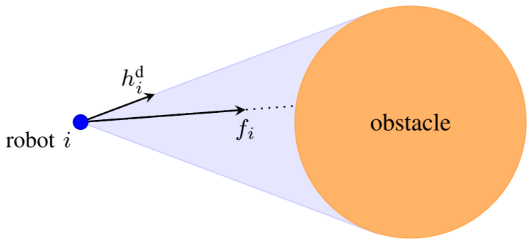

Figure 7: Avoiding a circular obstacle with a circle region.

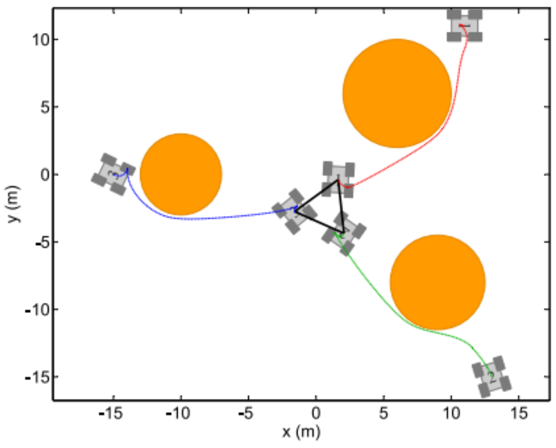

Figure 8: Illustration of robotic coordination with obstacle avoidance.

References

[1] M. M. Zavlanos, M. B. Egerstedt, and G. J. Pappas, "Graph-theoretic connectivity control of mobile robot networks," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1525–1540, 2011.

[2] F. Bullo, *Lectures on network systems*. Kindle Direct Publishing, http://motion.me.ucsb.edu/book-lns/, 2022.

[3] M. Mesbahi and M. Egerstedt, *Graph theoretic methods in multiagent networks*. Princeton University Press, 2010, vol. 33.

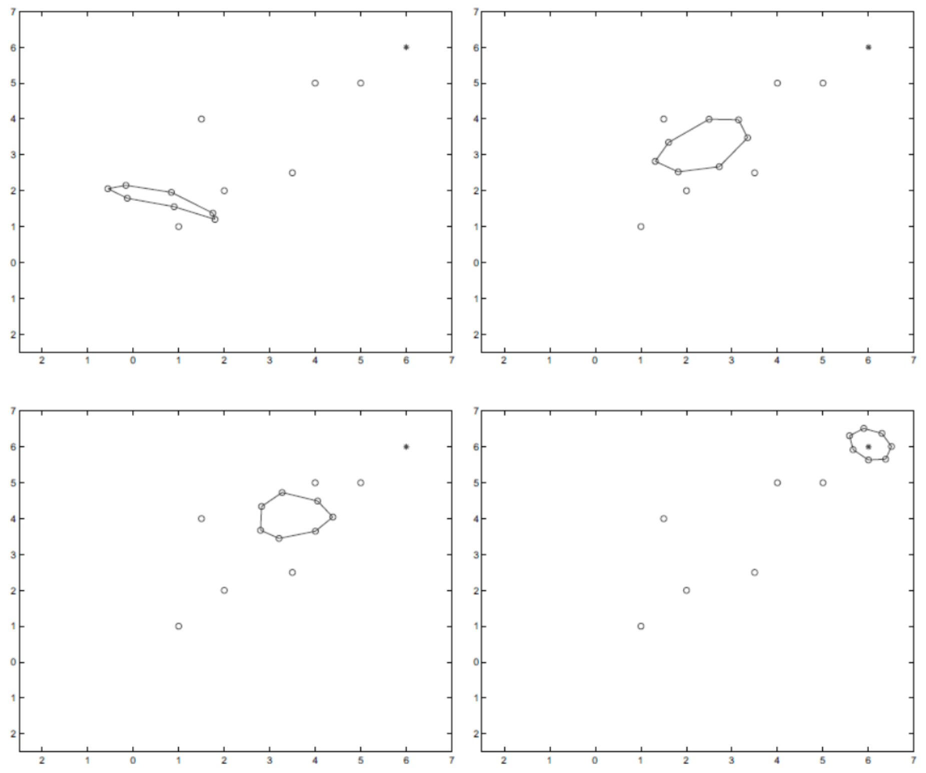
Figure 9: Formation movement while avoiding obstacles.