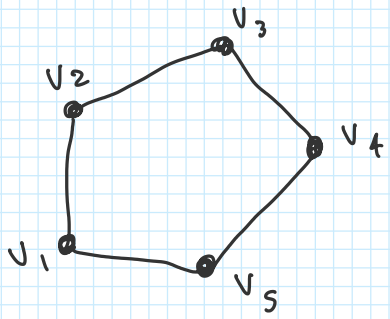
Thursday, April 24, 2025 9:24 PM

Q1)

gives:



where:

$$G = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix} \leftarrow \text{symmetric}$$

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

and:

and:

$$\vec{X}_0 = \begin{bmatrix} -4 & -2 & 1 & 7 & 5 \\ -5 & 2 & 6 & 0 & -6 \end{bmatrix}^T$$

$$\vec{K} = \begin{bmatrix} 0 & 0 & 1 & 0 & -1 \\ -1 & 1 & 0 & 0 & -1 \end{bmatrix}^T$$

Formation Protocol:

- $\dot{\psi}_i(t) = -\sum_{j \in N_{f,i}} \psi_i(t) - \psi_j(t)$
- $\psi_i(t) = x_i(t) - K_i$
- $\dot{x}_i(t) = -\sum_{j \in N_{f,i}} (x_i(t) - x_j(t)) - (K_i - K_j)$

double Integrator dynamics:

- $\dot{x}_i = v_i$
- $\dot{v}_i = u_i$

Then:

$$\dot{x}_i = v_i$$

$$\dot{v}_i = u_i$$

We can than say:

$$\dot{Z}_i = \begin{bmatrix} \dot{x}_i \\ \dot{v}_i \end{bmatrix} = \begin{bmatrix} v_i \\ k \sum_{j \in N_i} a_{ij}(x_j - x_i) + k\gamma \sum_{j \in N_i} a_{ij}(v_j - v_i) \end{bmatrix}$$

From lecture, we can stack the vectors and find the following state-space form of the dynamics:

From lecture, we can stack the vectors and find the following state-space form of the dynamics:

$$\dot{\vec{z}} = (I_N \otimes F - kL \otimes \vec{b}\vec{g})\vec{z}$$

For formation control:

$$\dot{x}_i = v_i = -\sum_{j \in N_{f,i}} (x_i - x_j) - (k_i - k_j)$$

Then :

$$\dot{z}_i = \begin{bmatrix} \dot{x}_i \\ \dot{v}_i \end{bmatrix} = \begin{bmatrix} -\sum_{j \in N_{f,i}} (x_i - x_j) - (k_i - k_j) \\ k \sum_{j \in N_i} a_{ij}(x_j - x_i) + k\gamma \sum_{j \in N_i} a_{ij}(v_j - v_i) \end{bmatrix}$$

We can define the neighborhood set for the given graph as follows:

$$N = \begin{bmatrix} 2 & 5 \\ 1 & 3 \\ 2 & 4 \\ 3 & 5 \\ 1 & 4 \end{bmatrix}$$

Finally, we can extend the augmented state vector to include the translation dynamics:

$$\dot{z}_i = \begin{bmatrix} \dot{x}_i \\ \dot{v}_i \\ \dot{\psi}_i \end{bmatrix} = \begin{bmatrix} -\sum_{j \in N_{f,i}} (x_i - x_j) - (k_i - k_j) \\ k \sum_{j \in N_i} a_{ij}(x_j - x_i) + k\gamma \sum_{j \in N_i} a_{ij}(v_j - v_i) \\ -\sum_{j \in N_i} \psi_i - \psi_j \end{bmatrix}$$

$$\dot{Z}_i = \begin{bmatrix} \dot{x}_i \\ \dot{v}_i \\ \dot{\psi}_i \end{bmatrix} = \begin{bmatrix} -\sum_{j \in N_{f,i}} (x_i - x_j) - (k_i - k_j) \\ \kappa \sum_{j \in N_i} a_{ij}(x_j - x_i) + \kappa\gamma \sum_{j \in N_i} a_{ij}(v_j - v_i) \\ \dot{x}_i \end{bmatrix}.$$

Can be computed *after* integration, but this makes it easy to compute everything at once

Graphs of each state :

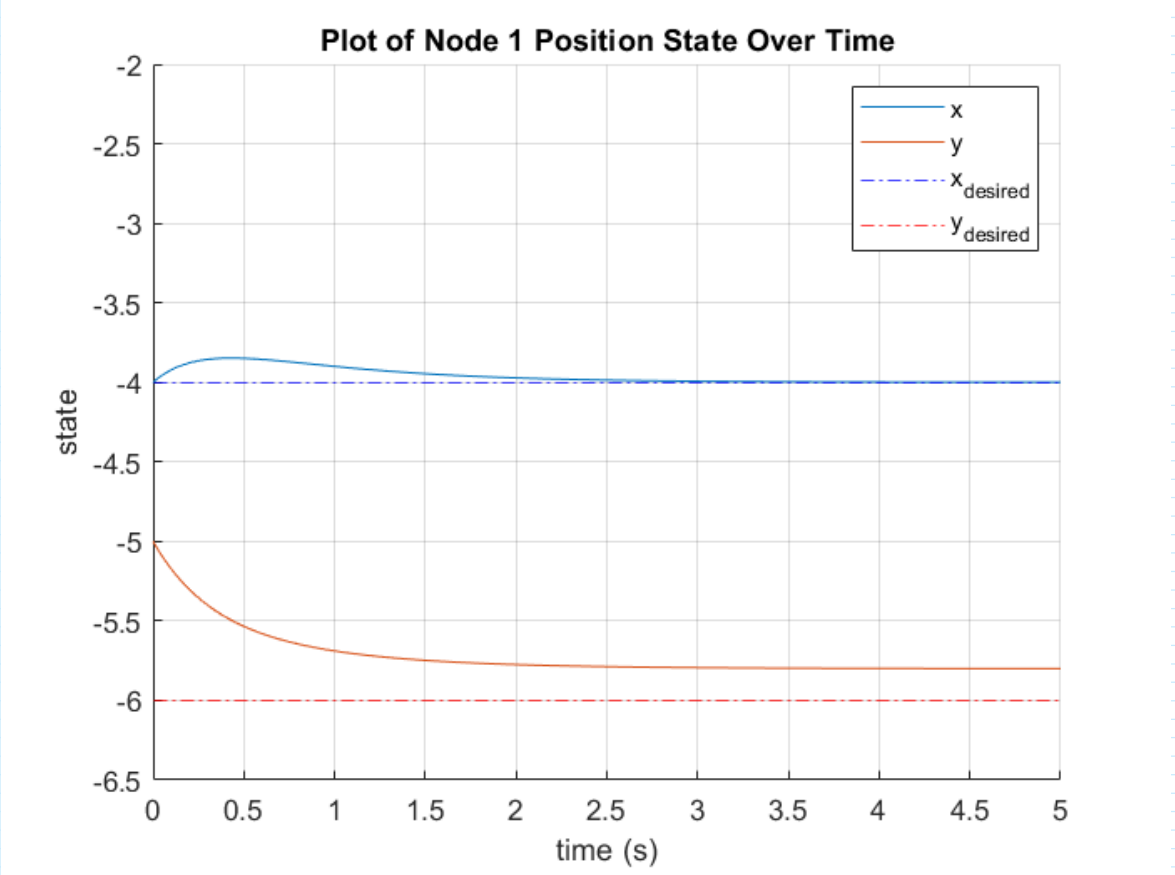
Using:

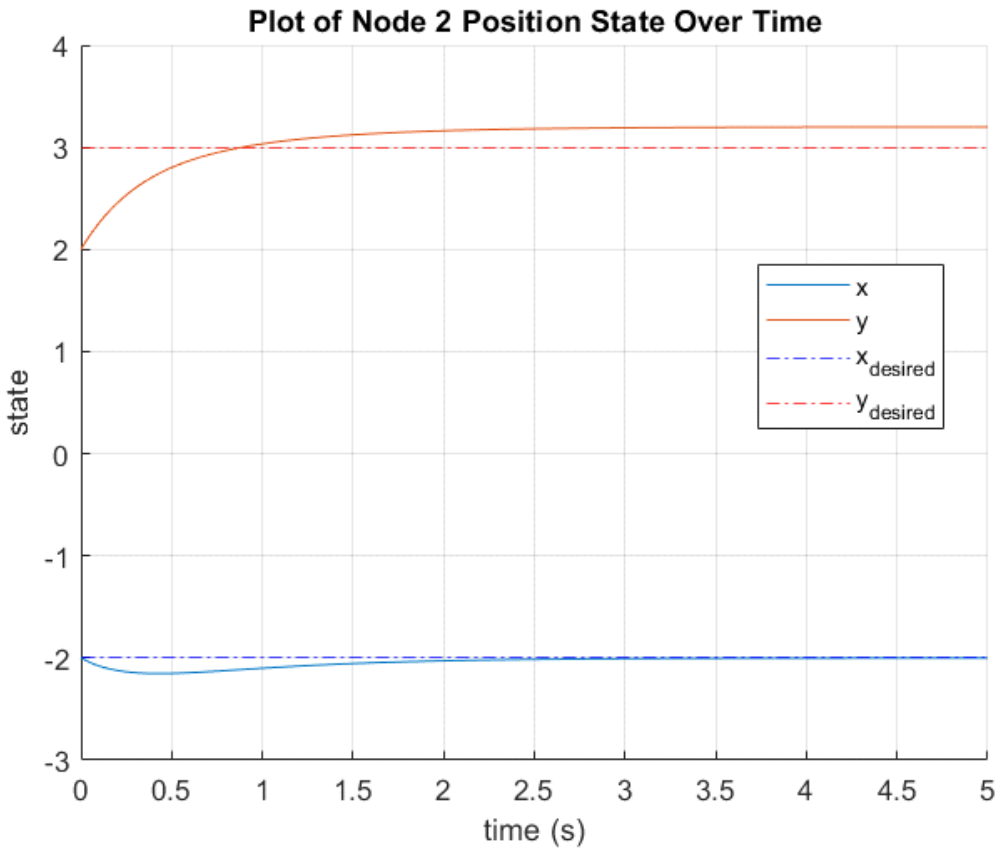
$k = 0.01$

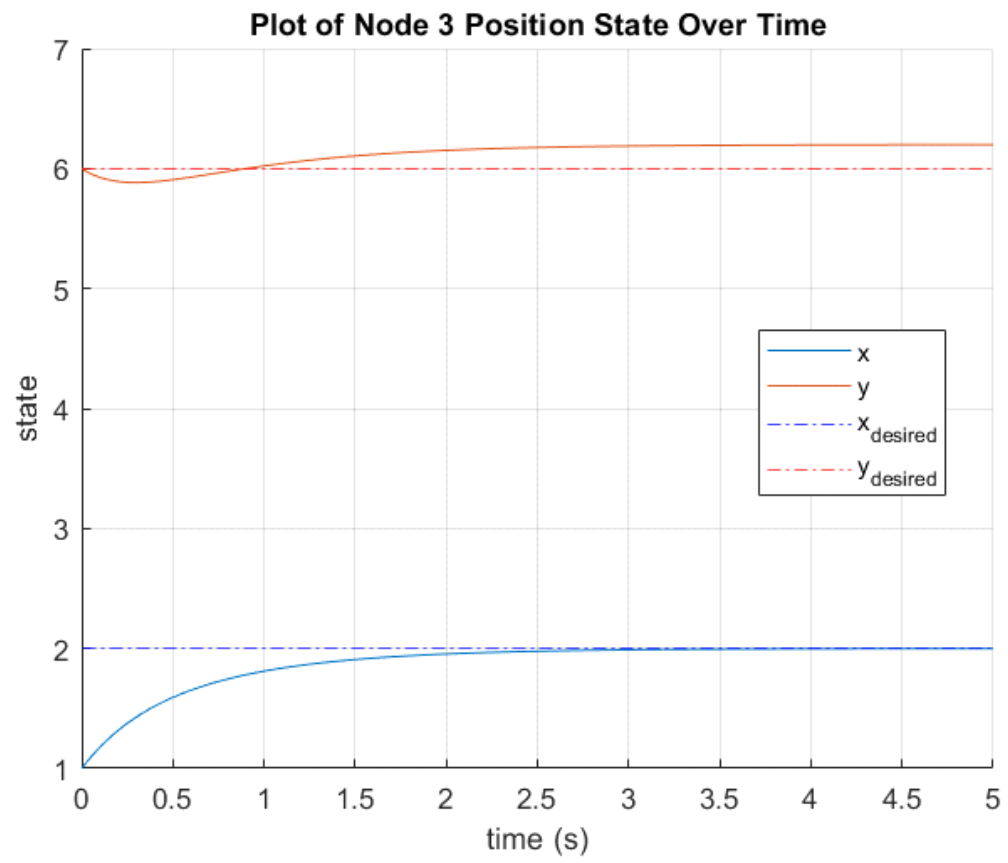
$\gamma = 100$

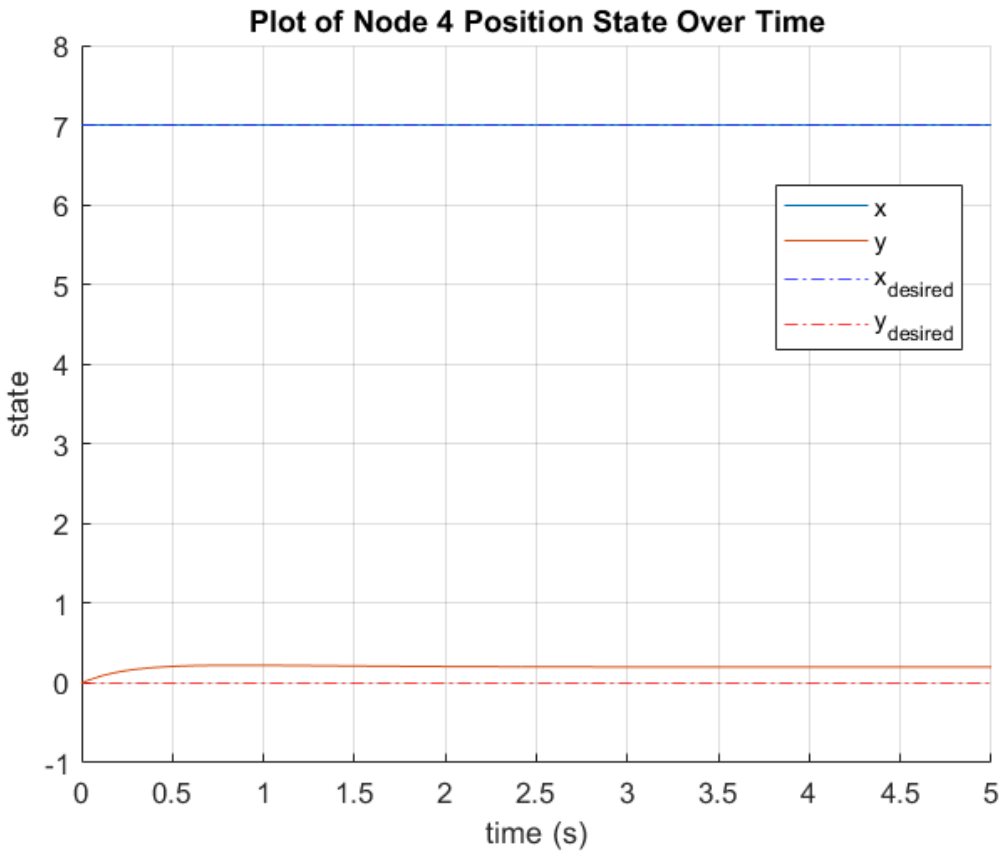
And

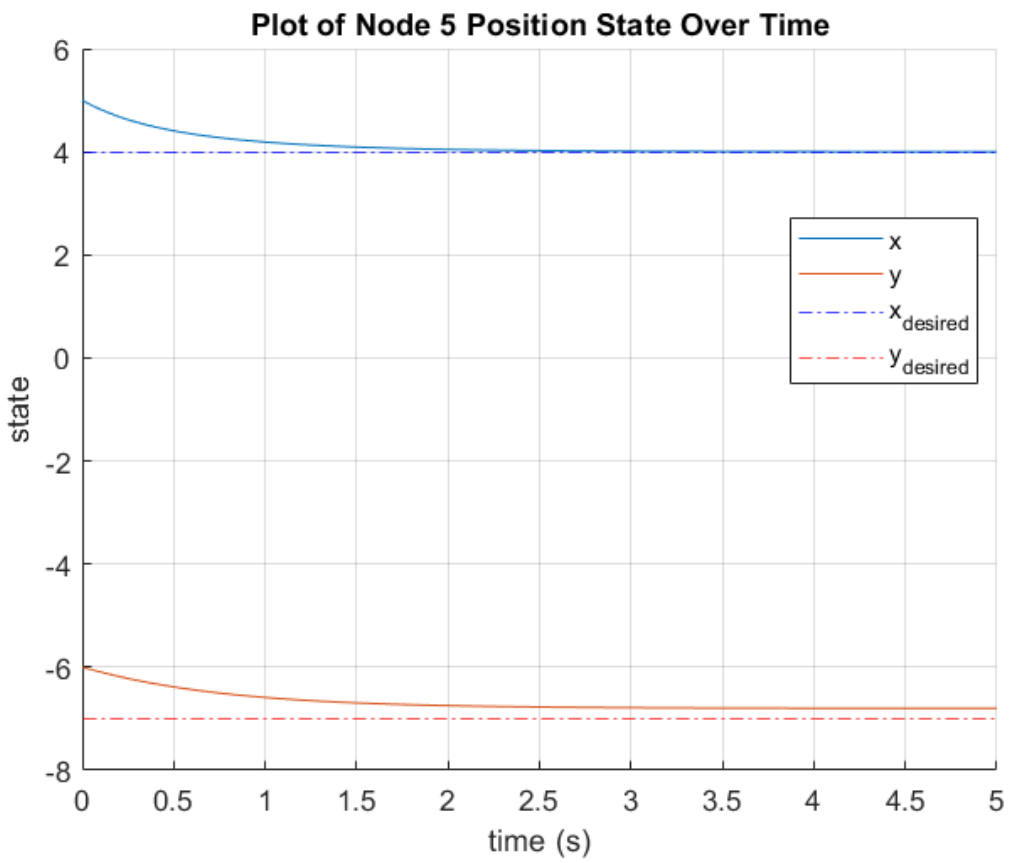
$$\vec{x}_{des} = \vec{x}_0 + \vec{k}$$

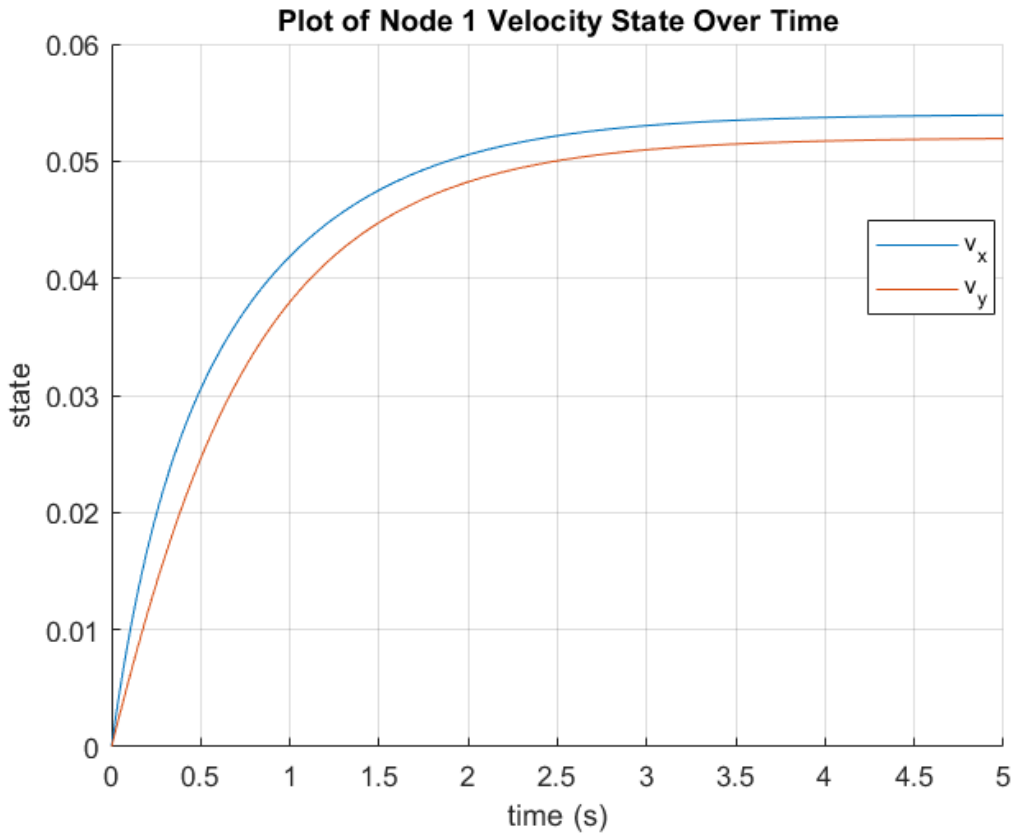


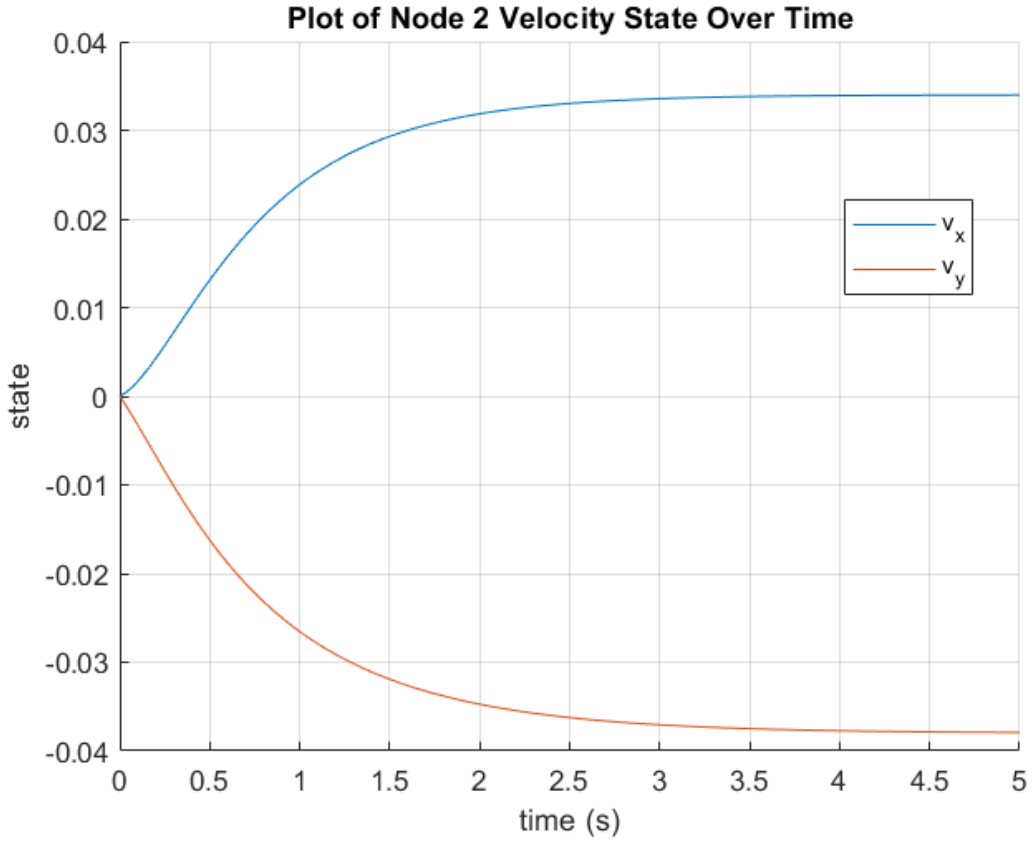

Plot of Node 2 Position State Over Time
state
time (s)
x
y
x desired
y desired

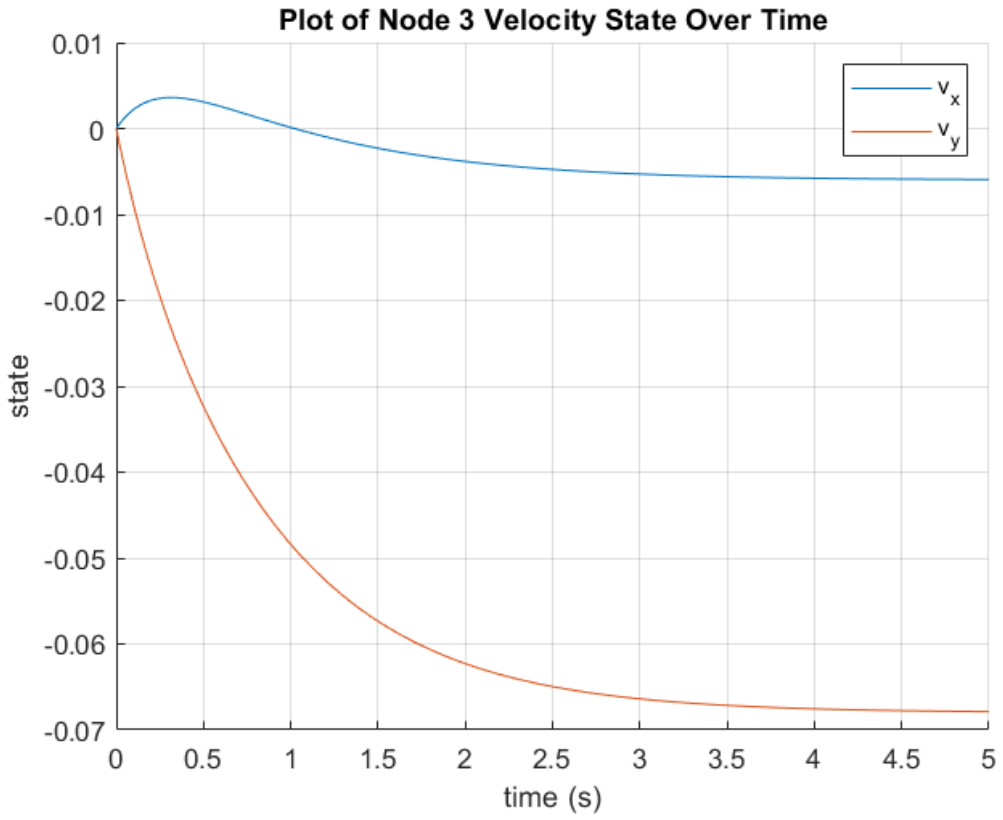

Plot of Node 3 Position State Over Time
state
time (s)
x
y
x desired
y desired

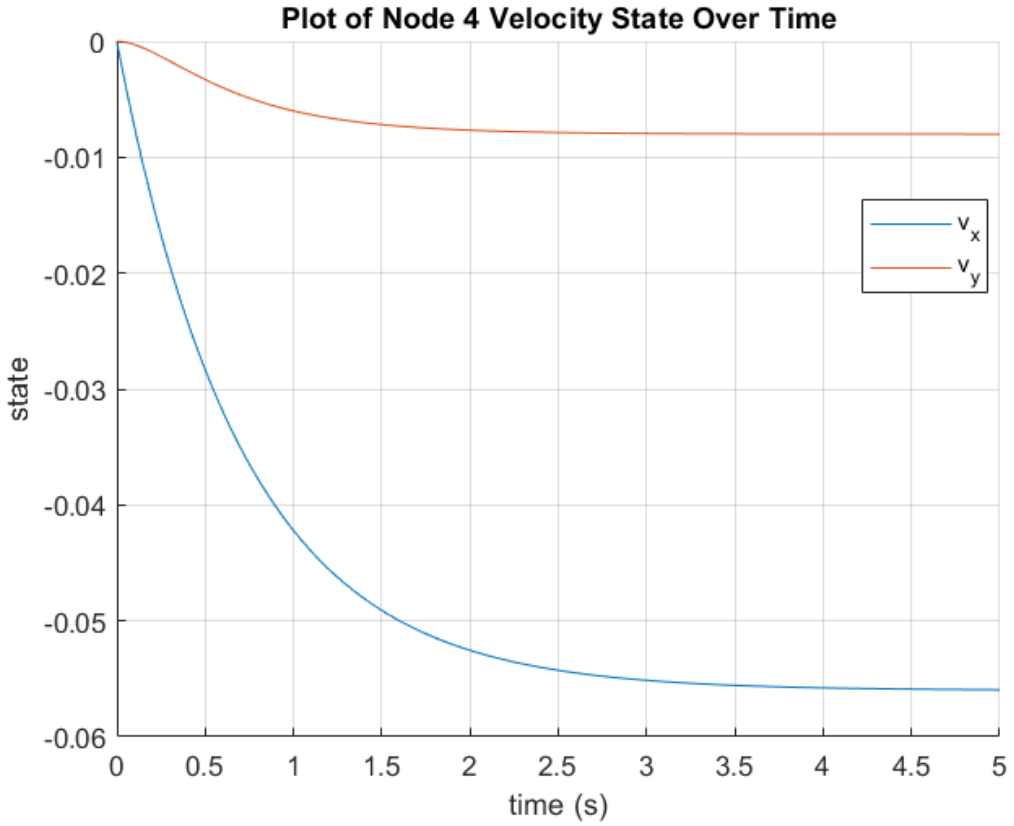

Plot of Node 4 Position State Over Time
state
time (s)
x
y
x desired
y desired

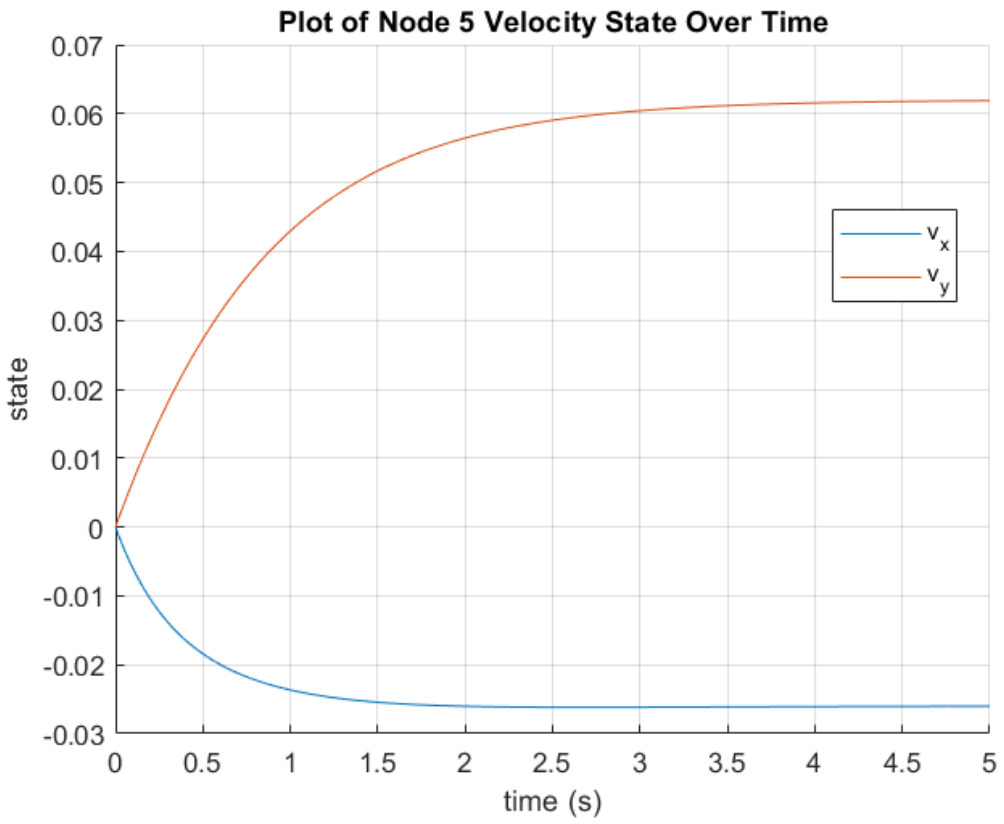

Plot of Node 5 Position State Over Time
state
time (s)
x
y
x desired
y desired

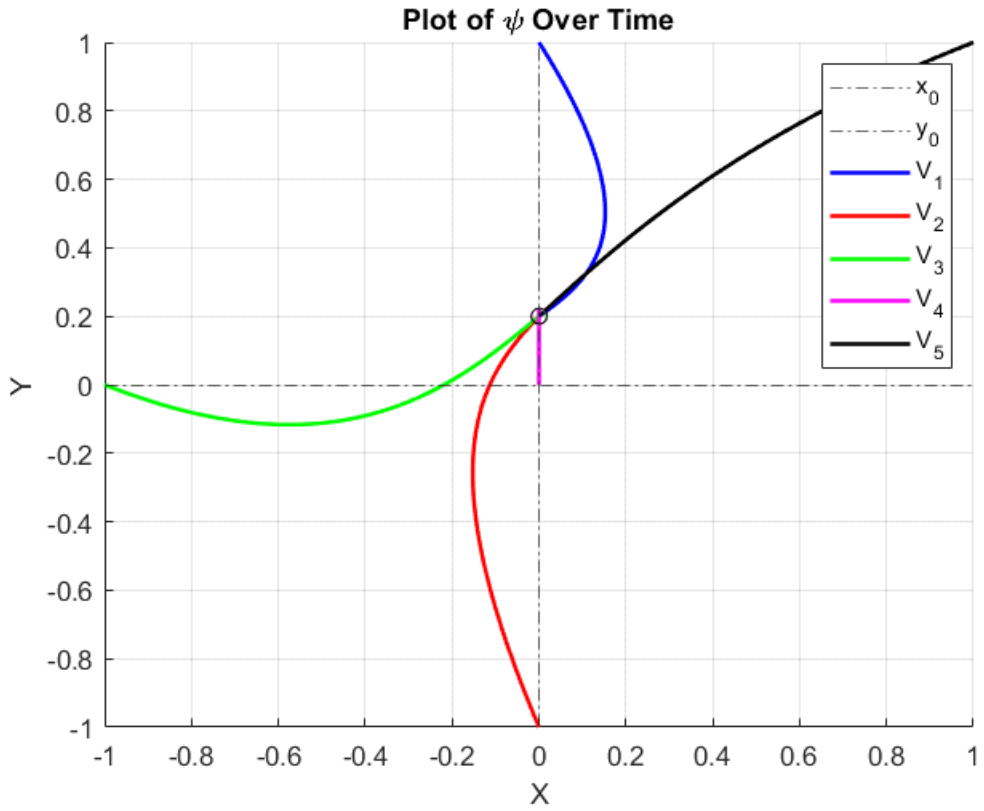

Plot of Node 1 Velocity State Over Time
state
time (s)
v_x
v_y
0.06
0.05
0.04
0.03
0.02
0.01
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

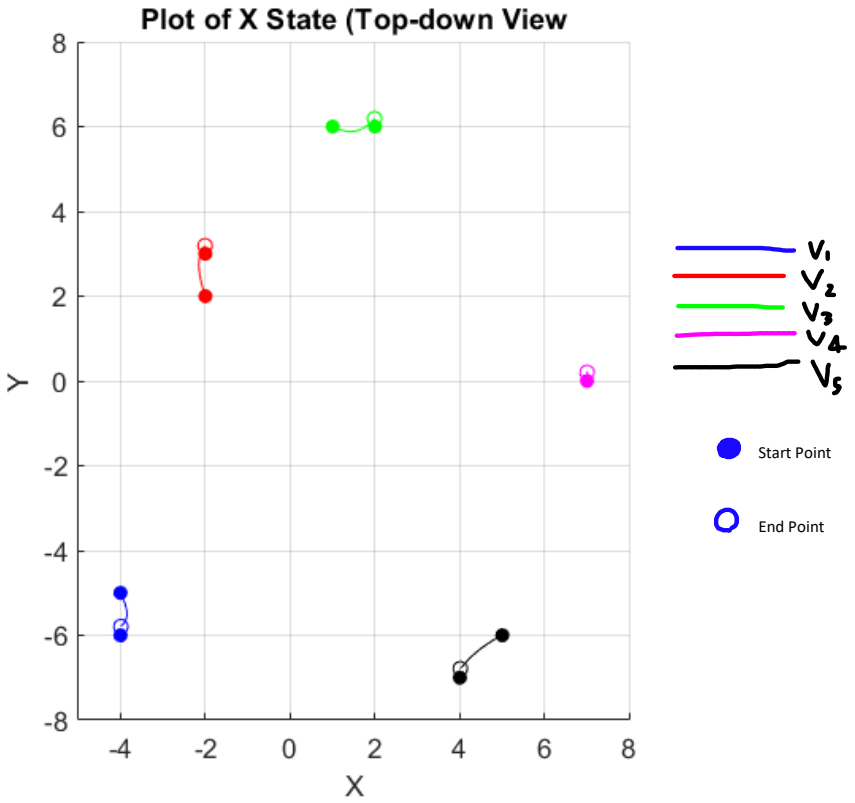

Plot of Node 2 Velocity State Over Time
state
time (s)
v_x
v_y
0.04
0.03
0.02
0.01
0
-0.01
-0.02
-0.03
-0.04
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5


Plot of Node 3 Velocity State Over Time
0.01
0
-0.01
-0.02
-0.03
-0.04
-0.05
-0.06
-0.07
state
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
time (s)
v_x
v_y


Plot of Node 4 Velocity State Over Time
0
-0.01
-0.02
-0.03
-0.04
-0.05
-0.06
state
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
time (s)
v_x
v_y


Plot of Node 5 Velocity State Over Time
state
time (s)
v_x
v_y


Plot of ψ Over Time
Y
X
x_0
y_0
V_1
V_2
V_3
V_4
V_5



Hmm, need to re-evaluate - implementation does not work for different k values…

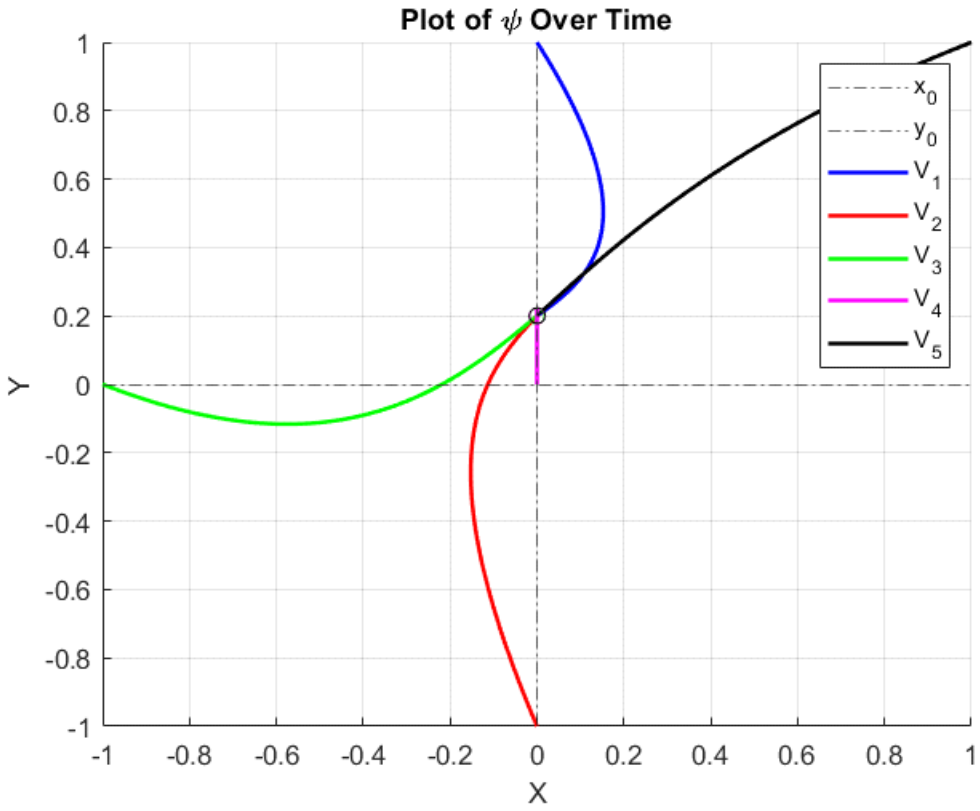
Attempt 2 :

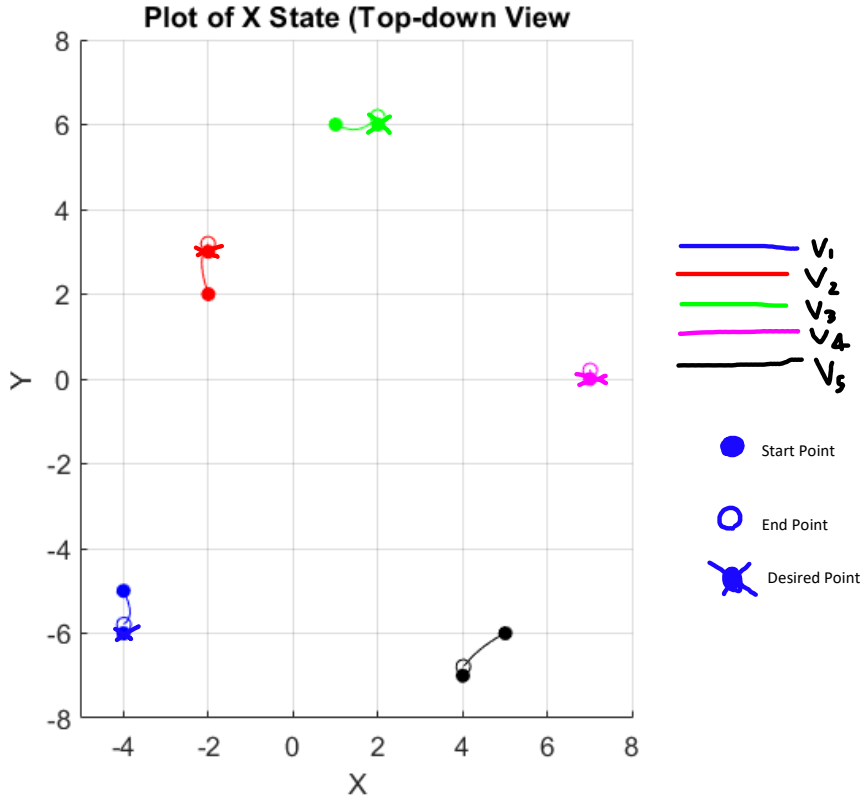
Propagating only formation dynamics, we see the following:

$$\dot{\Sigma}_i = \begin{bmatrix} -\sum_{j \in N_i} (x_i - x_j) - (k_i - k_j) \\ \dot{x} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{\psi} \end{bmatrix}$$

Given:

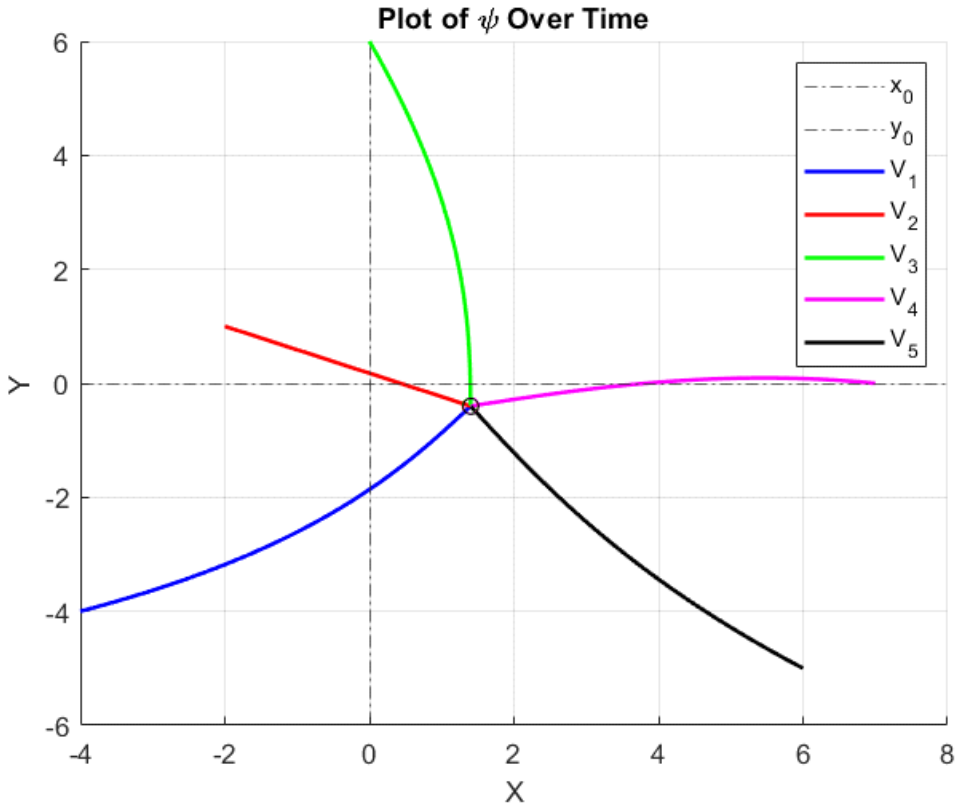
$$\vec{x}_{des} = \vec{x}_0 + \vec{k}$$

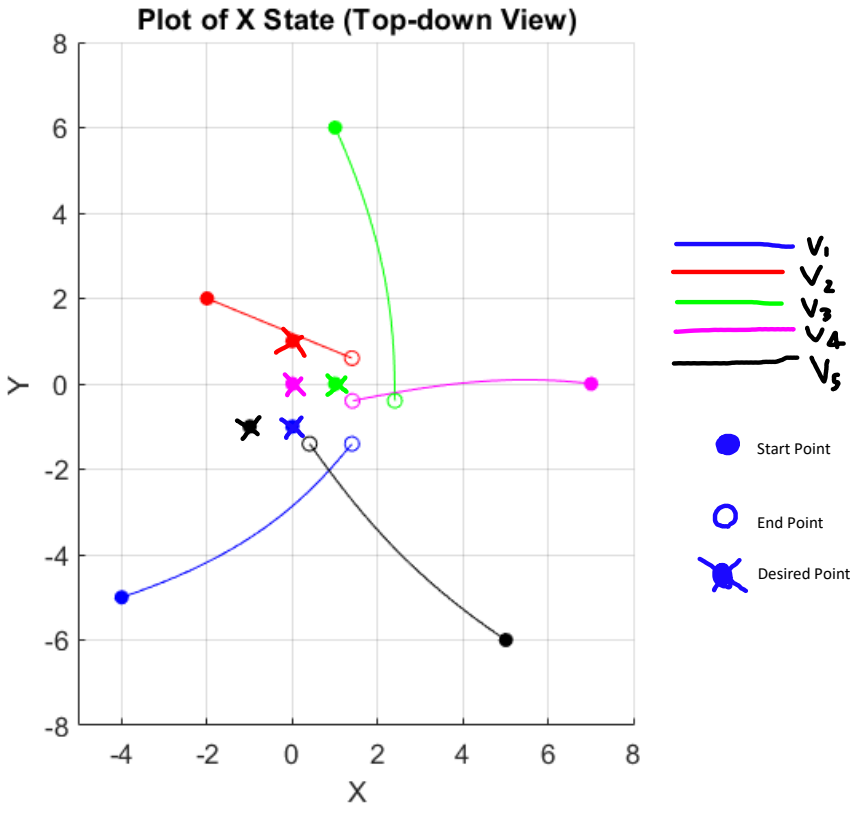

Plot of ψ Over Time
X
Y
x_0
y_0
V_1
V_2
V_3
V_4
V_5


Plot of X State (Top-down View
X
Y
V_1
V_2
V_3
V_4
V_5
Start Point
End Point
Desired Point

Given:

$\vec{X}_{des} = \vec{K}$





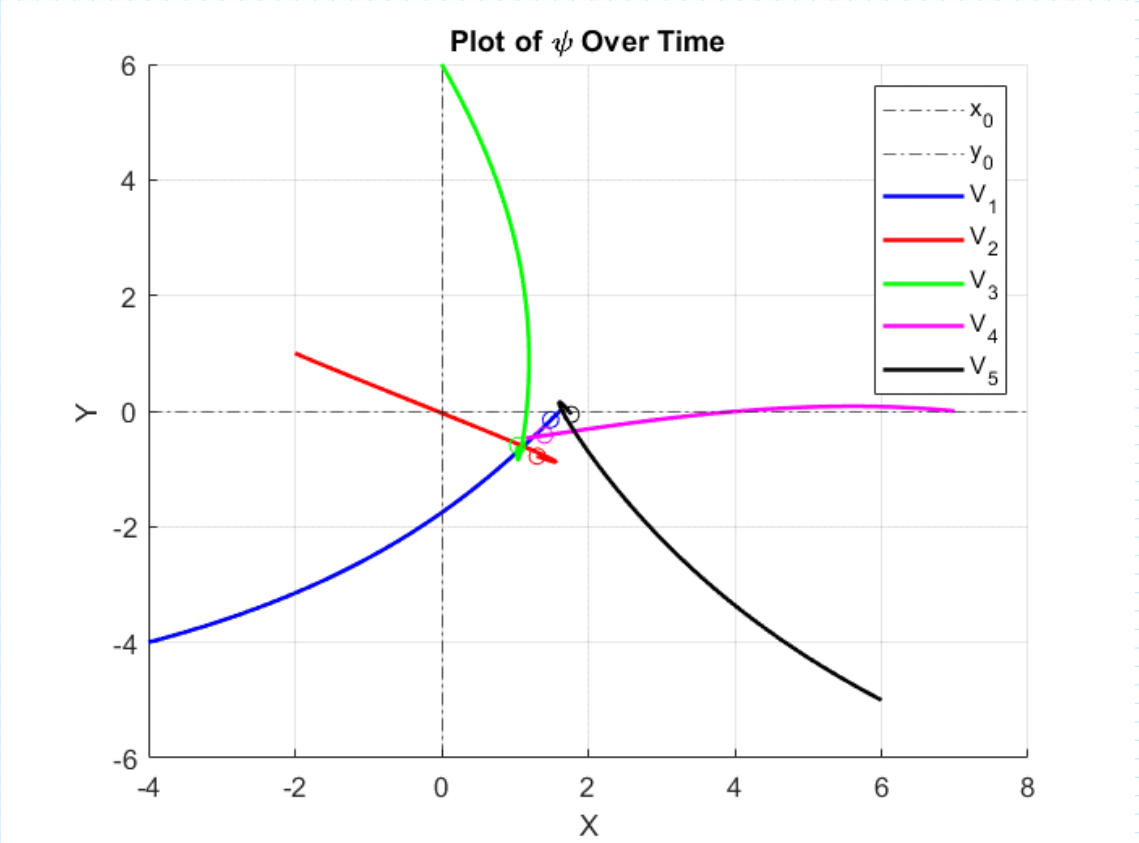
Just propagating formation doesn't work! (unless I'm messing it up…) - need to use double integrator dynamics like Dr. Otte said.:

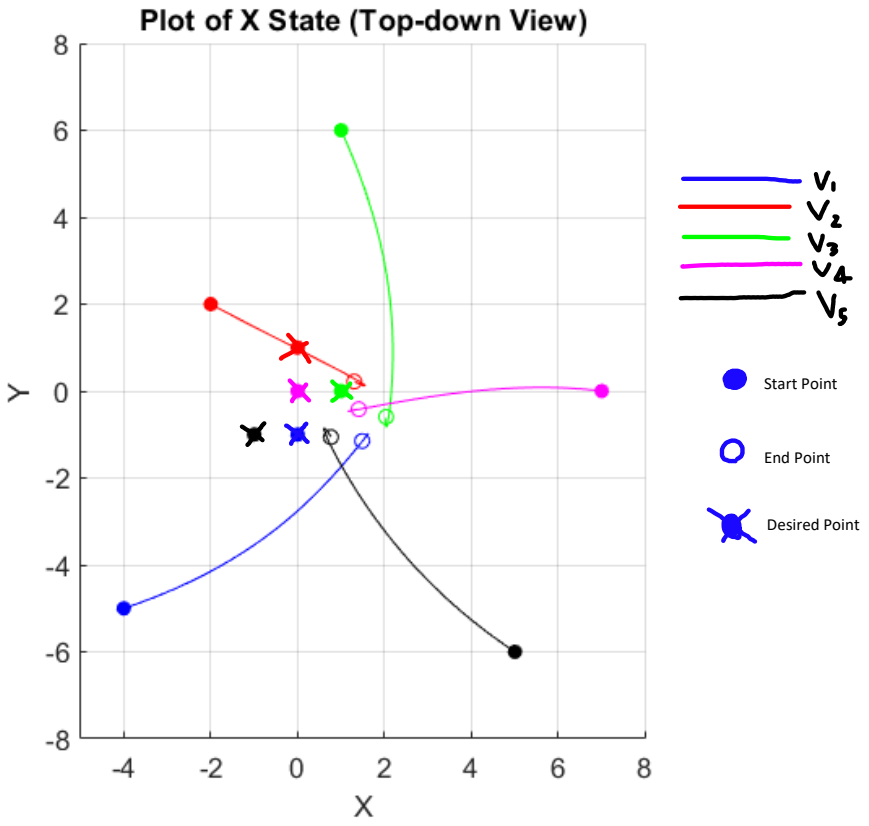
Attempt 3:

Given:

$$\vec{x}_{des} = \vec{k}$$

$$\dot{z}_i = \begin{bmatrix} \dot{x} \\ \dot{v} \\ \dot{\psi} \end{bmatrix}_i = \begin{bmatrix} -\sum_{j \in N_i} (x_i - x_j) - (k_i - k_j) + v_i \\ K \cdot \sum_{j \in N_i} a_{ij}(x_j - x_i) + k\gamma \sum_{j \in N_i} a_{ij}(v_j - v_i) \\ \dot{x}_i \end{bmatrix}$$





K and gamma gains just make it more unstable - this implementation is wrong!

Attempt 4 (FINAL):

From iteration, the following gains were found:

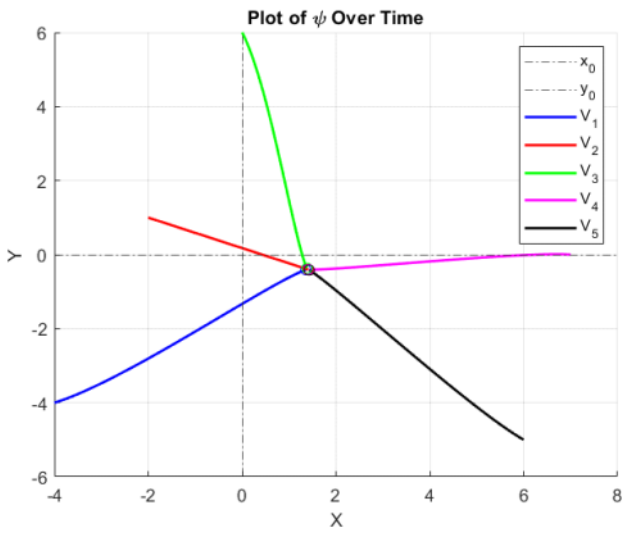
$K = 2.5$

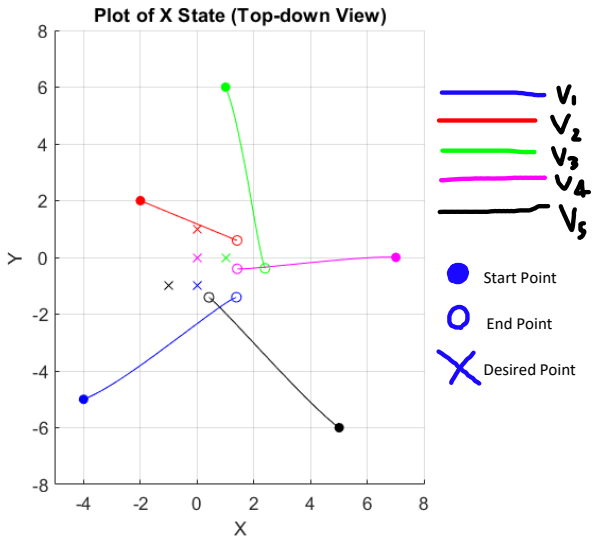
$\gamma = 0.7$

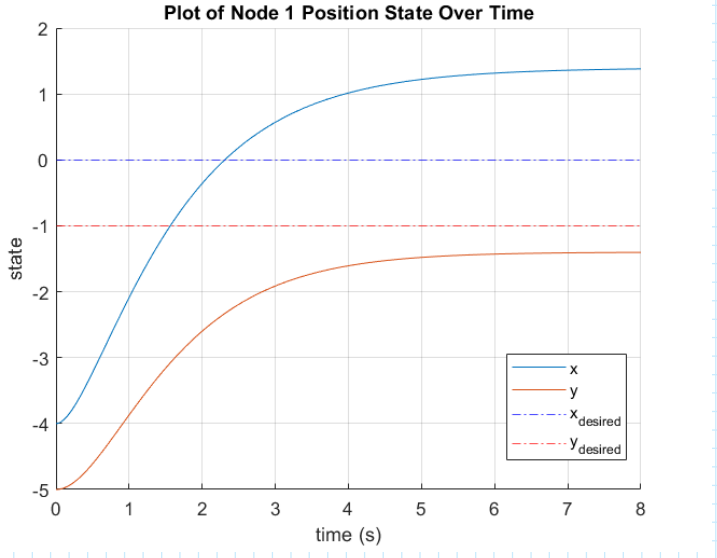
and using:

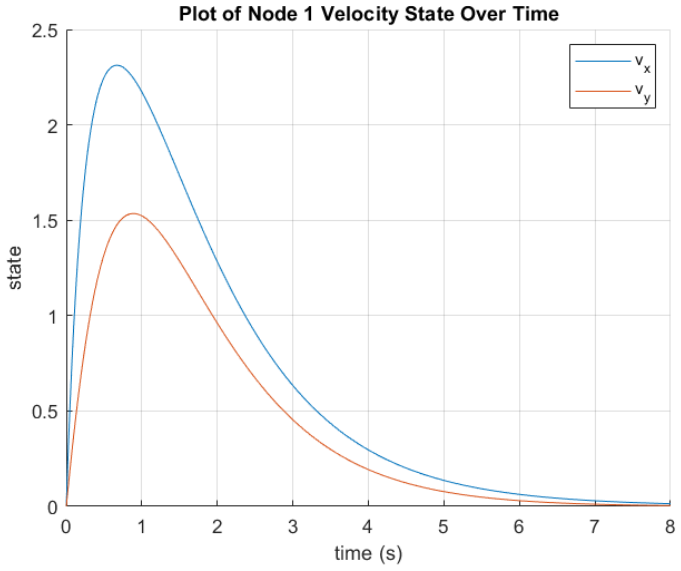
$$\dot{Z}_i = \begin{bmatrix} \dot{x}_i \\ \dot{v}_i \end{bmatrix} = \begin{bmatrix} v_i \\ -\sum_{j \in N_i} (x_i - x_j) - (k_i - k_j) + K_\gamma \sum_{j \in N_i} a_{ij}(v_j - v_i) \end{bmatrix}$$

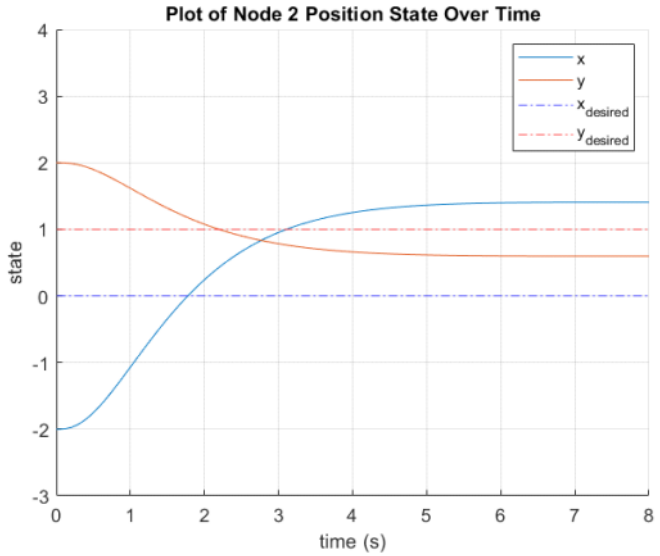
for $\vec{x}_{des} = \vec{k}$

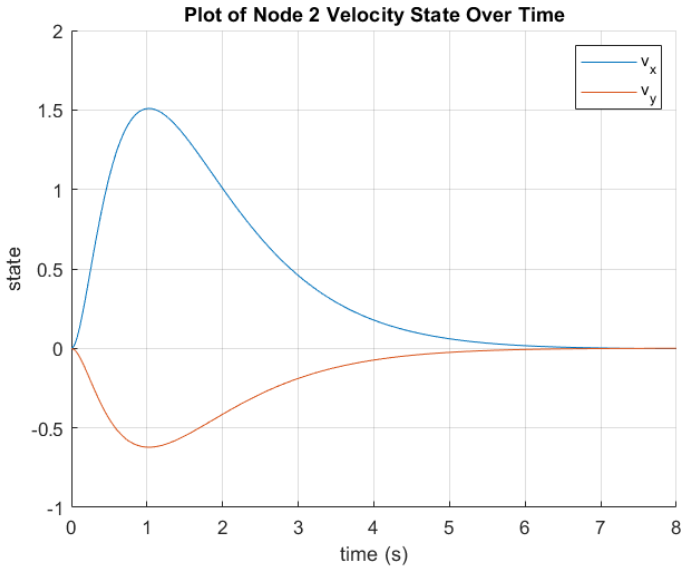

Plot of ψ Over Time
X
Y
x_0
y_0
V_1
V_2
V_3
V_4
V_5

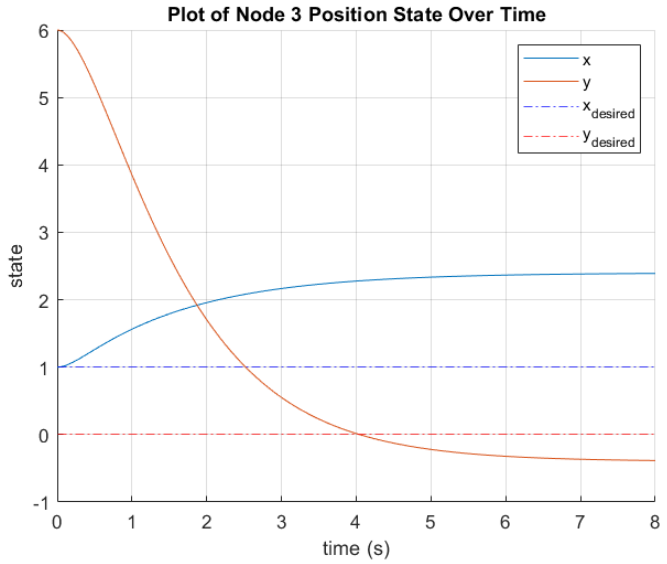

Plot of X State (Top-down View)
X
Y
V_1
V_2
V_3
V_4
V_5
Start Point
End Point
Desired Point

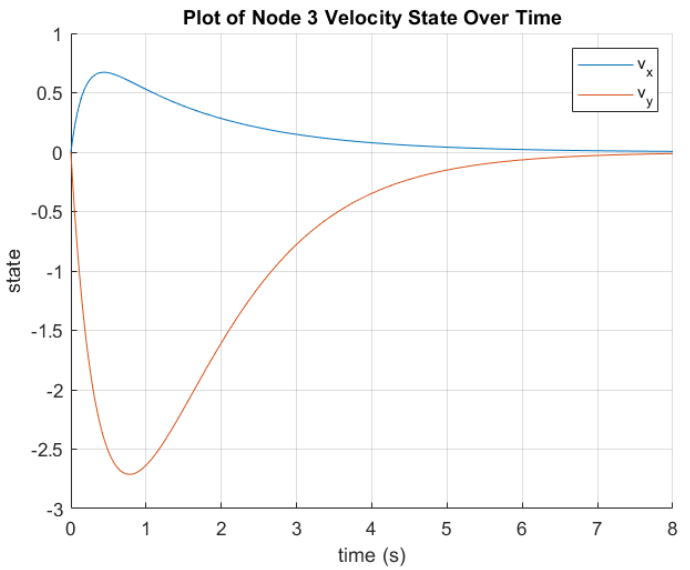

Plot of Node 1 Position State Over Time
time (s)
state
x
y
$x_{desired}$
$y_{desired}$

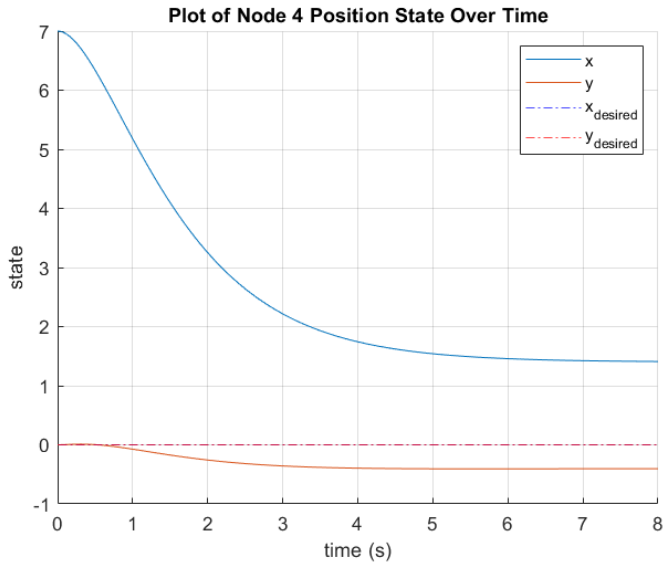

Plot of Node 1 Velocity State Over Time
time (s)
state
v_x
v_y

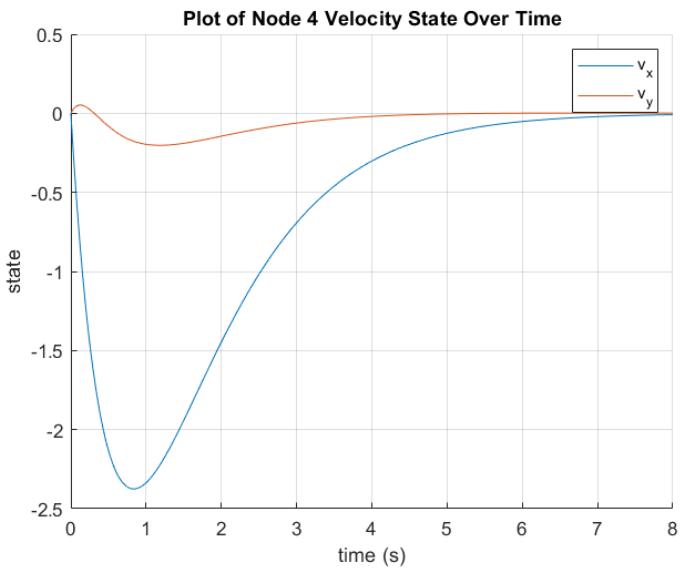

Plot of Node 2 Position State Over Time
x
y
x_desired
y_desired
state
time (s)


Plot of Node 2 Velocity State Over Time
v_x
v_y
state
time (s)


Plot of Node 3 Position State Over Time
x
y
x_desired
y_desired
state
time (s)

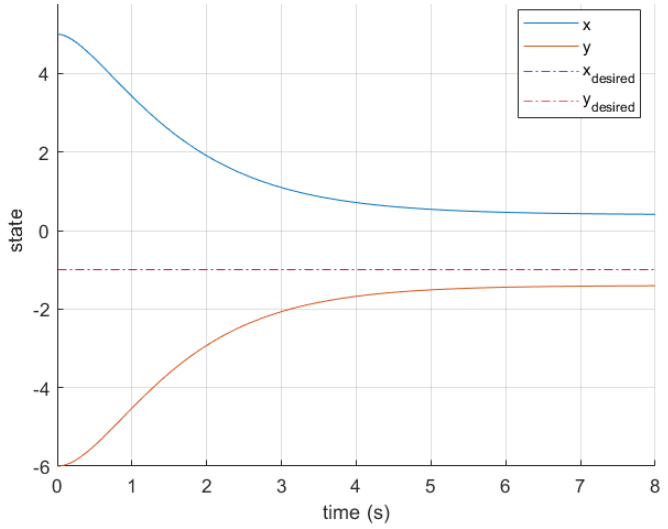

Plot of Node 3 Velocity State Over Time
v_x
v_y
state
time (s)

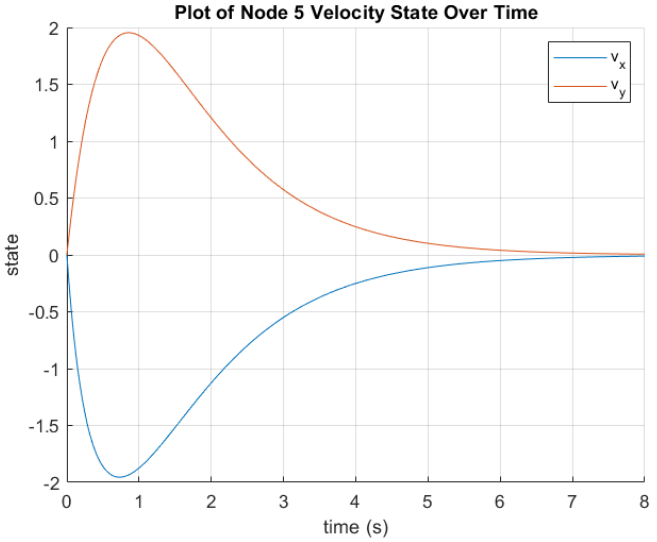

Plot of Node 4 Position State Over Time
x
y
x_desired
y_desired
state
time (s)


Plot of Node 4 Velocity State Over Time
v_x
v_y
state
time (s)

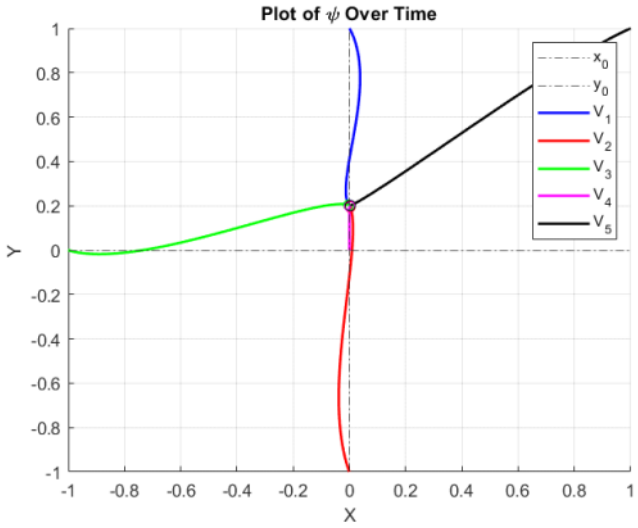
Plot of Node 5 Position State Over Time
x
y
x_desired

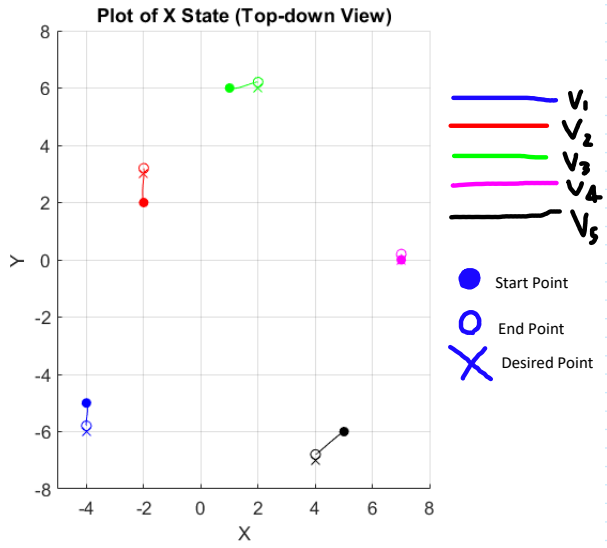
Plot of Node 5 Velocity State Over Time
v_x
v_y

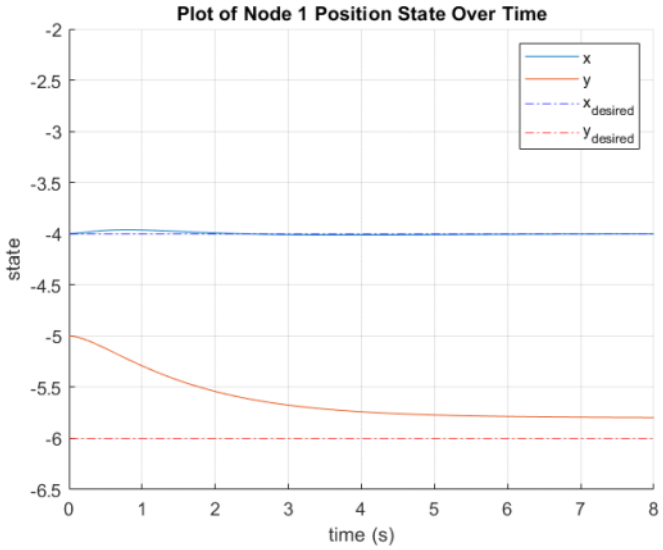


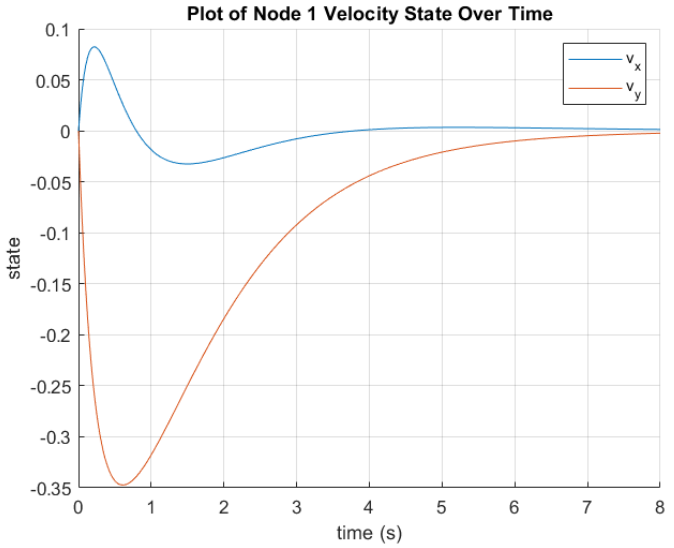


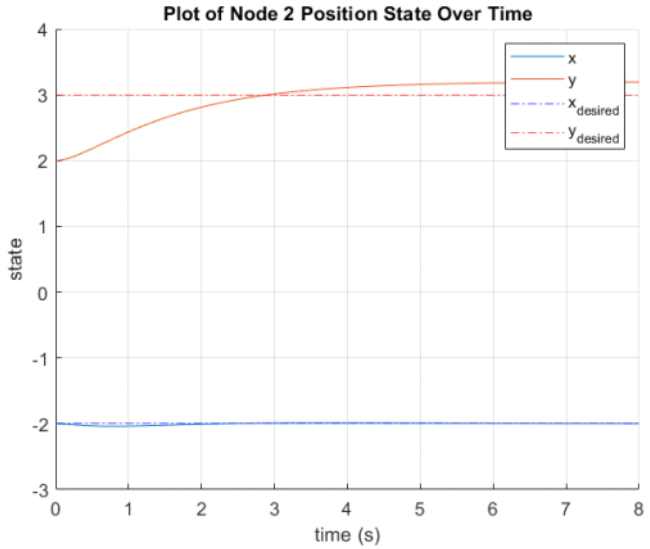
for $\vec{x}_{des} = \vec{x}_0 + \vec{k}$

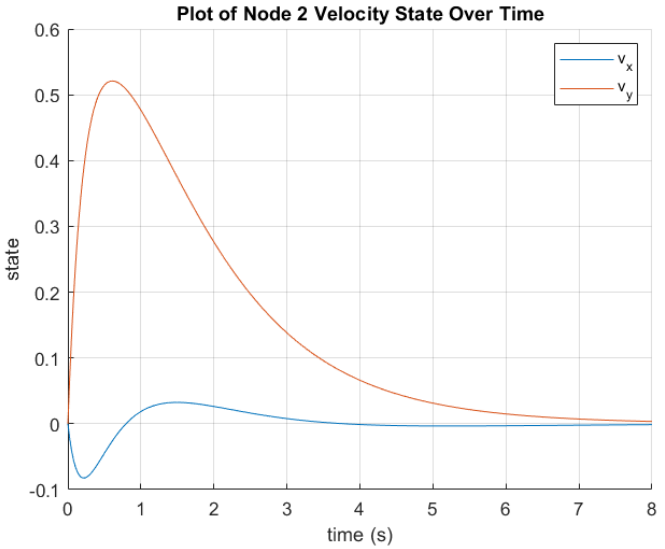


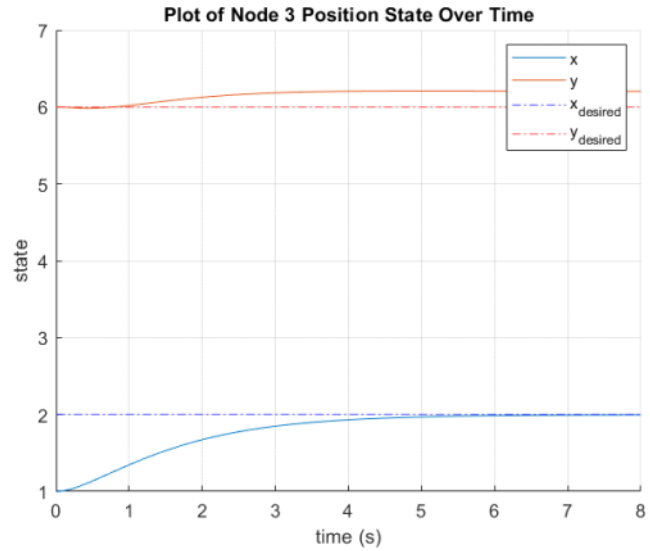


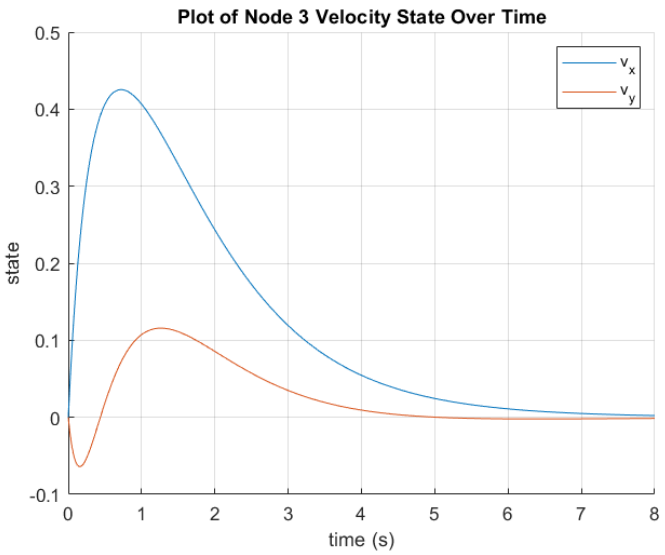

Plot of Node 1 Position State Over Time
x
y
x_desired
y_desired
state
time (s)

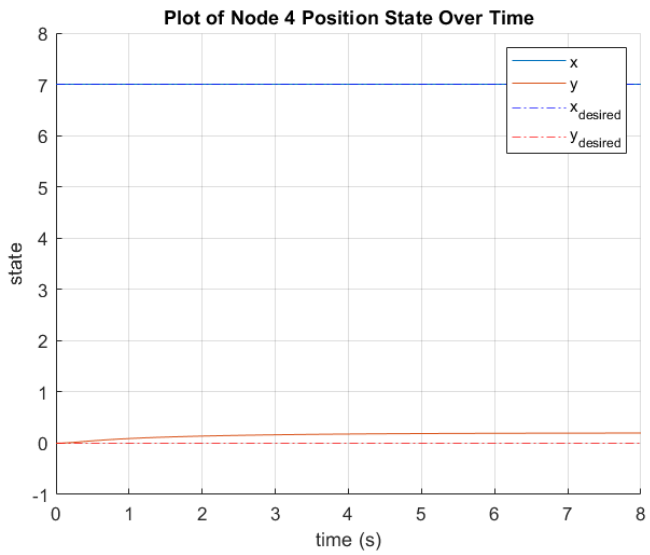

Plot of Node 1 Velocity State Over Time
v_x
v_y
state
time (s)

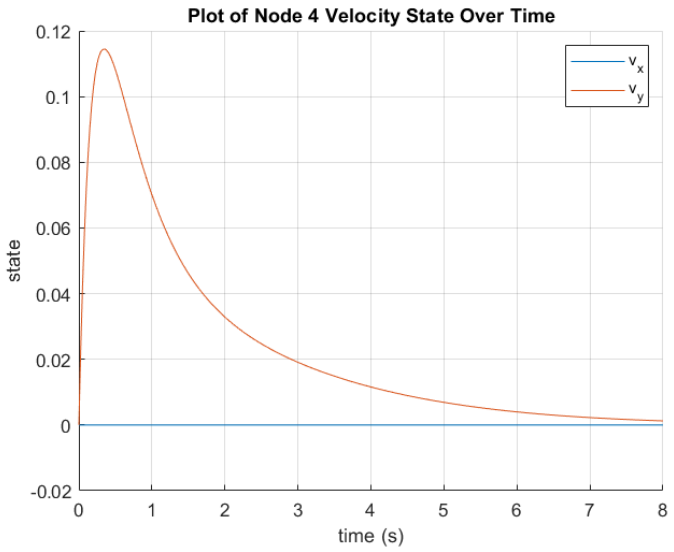

Plot of Node 2 Position State Over Time
x
y
x_desired
y_desired
state
time (s)

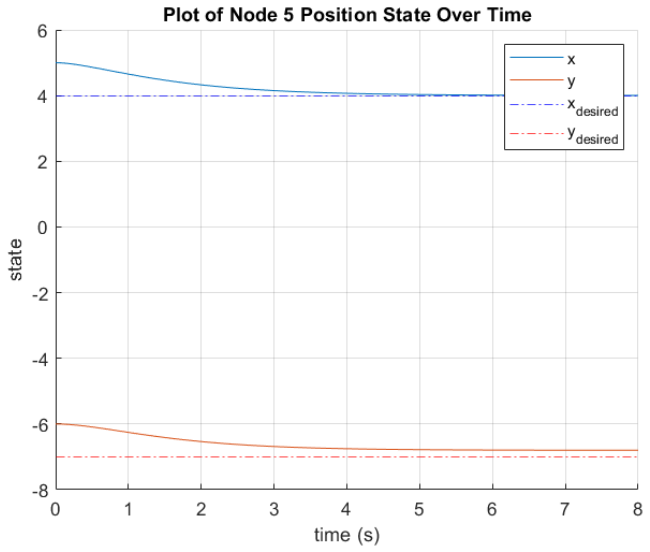

Plot of Node 2 Velocity State Over Time
v_x
v_y
state
time (s)

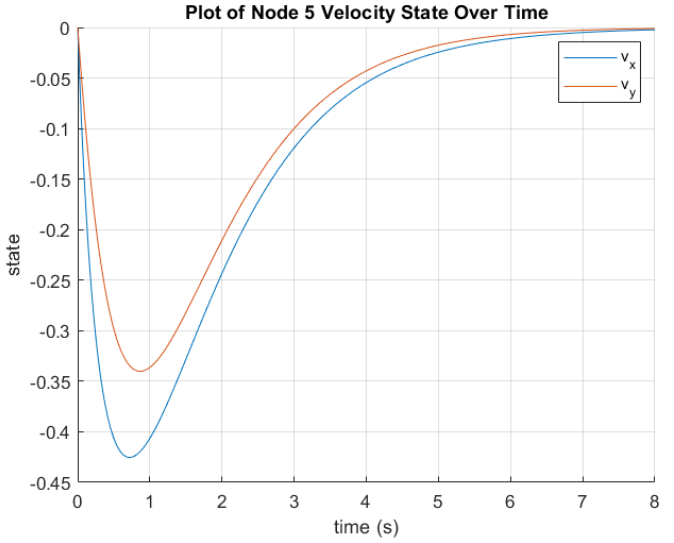

Plot of Node 3 Position State Over Time
x
y
x_desired
y_desired
state
time (s)


Plot of Node 3 Velocity State Over Time
v_x
v_y
state
time (s)









After modifying the code several times to try and get the desired output, I noticed a few things:

- Since I'm using ODE45 to propagate the state, the time step supplied greatly affects the speed at which the system approaches steady-state. However, the system approaches steady-state at the same time, every time. As in, a larger time step has less granularity in the discrete states generated, but the system still approaches steady-state at the same time. Of course, the propagated states produce a smoother plot when the timestep is smaller.
- For some reason, the nodes are unable to reach the desired position, and I'm not sure why. When I assume **k** is the relative x,y positions with respect to the nodes current position, the formation can almost reach their desired state, however there is a consistent offset in the y-position of each node, such that each of the nodes are always roughly 0.2 off in the y-direction from their desired state. When I assume **k** is the absolute target x,y position for the node,

each node again reaches a target position that is offset from the actual desired target position.

What's weird to me is that for both cases, the formation asymptotes to a clear x,y position, but it's not the desired **k**. I think it's because the nodes are adjusting based off of their neighbors position (not their global position), but I assumed that the formation protocol would drive the formation to their desired targets **k** always, and not just get close. Even if the time is increased to infinity, the nodes would not reach their desired **k**. I've tried a lot of different things to get **psi** to asymptote to 0 for each node, yet weirdly it seems to asymptote to another location that is not zero. Again, I think it is due to the formation protocol and the fact that the state is derived from the relative position to the nodes neighbors, but I'm not sure.

- After incorporating velocity into the system, we can see that the velocity will asymptote to 0 when the node approaches it's desired position.

Again, for some reason, I could not get the formation to approach the desired target for each **k**, regardless of the tuning parameters used for the velocity in the double integrator portion. From what I could tell, my implementation of the Formation protocol was correct, however there could be something wrong with it that I could not see. Again, the system itself was unaffected by the Double Integrator model, as it demonstrated the same asymptotic behavior when analyzing the single integrator dynamics!

Wednesday, April 30, 2025 11:20 PM

Q2)

A) An example of positive emergent behavior can be found when analyzing the multi-agent system employed by Waze. This is a little bit of a stretch, but Waze, the driving directions app, can be thought of as a multi-agent system, where uses are individual agents. Each user provides information about cops, accidents, traffic, ETC, and what emerges from the network is a system that can provide optimal routes to users without needing global information about the road

   Like I said, kind of a stretch but I think it can work! It's like if you had a set of individual agents that communicate to each other about certain features of their surroundings, and what emerges from each of the agents communicating this information is an optimal path through their environment

B) One example of neutral emergent behavior can be found when looking at the stock market. Each individual who invests in the stock market (an agent) influences it. Behaviors can emerge from this, such as repeated trends in the stock market or the indication of the economy doing well or doing poorly. This behavior isn't necessarily positive or negative, it just kind of emerges as a natural part of agent interaction

C) One example of negative emergent behavior can be found when investigating overfishing in bodies of water. Each agent (person fishing) is focused on collecting fish for themselves and has no information about any other person fishing. Because the agent is excluded from the state of the fish/other people fishing, the body of water loses its fish population as people overfish. In this case, the emergent behavior is that the fish depletes from the body of water, even though each person fishing is not aiming for that outcome, and only concerned about their fish intake.

Q3)

A) Three different meanings that roboticist could think of when they hear the word swarm could be:
   a. Just a very large group of robots working together
   b. A bunch of decentralized, individual robots in a group that are working to complete a high level task
   c. A network of robots that act independently yet a behavior emerges.

   Certainly, when I think of swarm I think of an incredibly large amount of robots working together

B) One difference between a robot team and robot swarm is that a robot team has the robots actively communicating with each other in order to accomplish a goal, and each robot has some idea of what the goal at hand might be, whereas in a robot swarm each robot has no real idea of what the goal of the swarm is, and only an idea of what the robot needs to do (not necessarily communicating with other robots), and through emergent behavior the swarm can accomplish that goal.

   Another difference between a robot team and a robot swarm is that robot swarms usually consist of many, many robots, whereas a robot team does not need to consist of many robots, and can in theory only require two or more robots.

C) One way that a robot team is similar to a robot swarm is the fact that teams and swarms both consist of multiple robots (duh). Another way that they are similar is that the use multiple robots to accomplish a specific goal that is not able to be accomplished by a single robot.