Abstract—The problem of decentralized control of multi-agent nonlinear systems is solved by introducing the concept of virtual agents to generate reference trajectories to be tracked by the actual agents. The tracking problem as an optimal control problem is formulated in the framework of approximate dynamic programming. Solutions are obtained using ‘single network adaptive critics’ and network weight update rule is discussed. Finally, the proposed theory is simulated on the network of several satellites to reach consensus on their attitude. Numerical results demonstrate the versatility and usefulness of the proposed approach.

Keywords—Decentralized control; Nonlinear multi-agent system; Optimal tracking control; Adaptive critics; Satellite formation control

I. INTRODUCTION

Decentralized control of multi-agent systems is one of the challenging problems in the control literature. The problem in designing such a controller is the issue of utilizing only local information of the neighbors to pursue some global behavior for the system, for example, formation control of a system of agents. The topic of consensus/formation control of linear agents is rich in literature. Some of the developments are listed here.

In [1] a method called behavior-based decentralized control is developed that performs the multi-agent control for network of agents with ring-wise communication topology by considering two desired behaviors: formation keeping and goal seeking. The role of the communication topology in the formation’s stability is shown in [2] through the appearance of the eigenvalues of Laplacian matrix of the communication graph in the dynamics of the formation. In [3] also, the stability of the formation/consensus control is shown to be related to the eigenvalues of the Laplacian matrix and for the double-integrator system, the authors of [3] have designed a controller based on the eigenvalues of this matrix. Minimization of some performance index to reach a consensus is the approach used in [4] and [5] for single integrator and double integrator dynamics, respectively. Many of the developed methods in the literature are applicable only to single or double integrator dynamics [1], [3]-[8]. In some other papers, the consensus protocol is developed for higher-order dynamics, but still limited to a multi-integrator structure, i.e. linear Brunovsky canonical form, for example in [9].

In practice, dynamics of the agents could be much more complicated and hence, developing a consensus protocol which guarantees the consensus for the agents of general linear dynamics is of importance. Studies in [10]-[12] consider this point by decomposing the eigenvalues of the Laplacian matrix, [3], and designing control gains such that the agents’ closed loop dynamics are stable. A different approach is selected in [13] to serve the goal by solving a linear matrix inequality which contains the Laplacian matrix.

Designing a decentralized control law for network of nonlinear heterogeneous dynamics is still an open problem in the literature. The developed methods in this regard mainly include the approaches of feedback linearization [1] and [20], and virtual structure [21]. In the feedback linearization based papers, the nonlinearity is cancelled through a suitable expression for control to end up with network of linear agents, while in a virtual structure design as in [21] one node is responsible to generate some desired signals for different agents to be tracked by them, hence, resulting in a centralized control in the sense that all the agents need to be able to communicate with that node/agent. For a specific communication topology, in [22] the virtual structure scheme is modified to make it decentralized for agents with some particular dynamics.

This paper’s contributions are in two areas. First, a scheme for formation/consensus control of nonlinear agents is developed called the virtual agent scheme which utilizes a network of virtual identical linear dynamics to be controlled using a decentralized control law and generate reference signals for the actual agents to be tracked by them. The second contribution is in developing an ‘adaptive critic’ based neurocontroller [19] for tracking which provides a comprehensive closed form solution though the solution is obtained offline.

Solving optimal tracking problems for nonlinear systems using adaptive critics has been investigated by researchers in [23]-[28]. In [23] the authors have developed a tracking
controller for the system whose input gain matrix, i.e. matrix $g(\cdot)$ in the state equation (1), is invertible.

$$x_{k+1} = f(x_k) + g(x_k)u_k$$  

(1)

In [24] the reference signal is limited to those which satisfy the dynamics of the system, i.e. denoting the reference signal by $r$, it needs to satisfy $r_{k+1} = f(r_k) + g(x_k)u_k^d$ where $u_k^d$ denotes the desired control and the state equation is given by (1). Developments in [25]-[28] solves the tracking problem for the systems of nonlinear Brunovsky canonical form.

The developed neurocontroller in this study is called Tracking SNAC for solving optimal tracking problem of nonlinear control-affine dynamics for tracking a signal whose given dynamics could be non-linear and arbitrary, hence, the limitations of the cited methods do not exist for this controller.

Rest of the paper is organized as follows: the problem formulation and defining the virtual agent scheme is given in section II. Afterward, a Single Network Adaptive Critic (SNAC) based neurocontroller, called Tracking SNAC, is developed for infinite-horizon optimal tracking of a reference signal in section III and is used for the formation control of the heterogeneous nonlinear agents in section IV. Numerical results are provided in section V, followed by conclusions in section VI.

II. PROBLEM FORMULATION

Assume a network of $N$ agents whose heterogeneous nonlinear dynamics are described by

$$\dot{x}_i(t) = f_i(x_i(t)) + g_i(x_i(t))u_i(t) \quad i = 1, 2, ..., N$$  

(2)

with initial conditions $x_i(0)$ where $x_i(t) \in \mathbb{R}^n$ and $u_i(t) \in \mathbb{R}^m$ are the state and the control vectors and $f_i(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^n$ and $g_i(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times m}$ are the system dynamics of the $i$th agent. The objective here is to design $N$ decentralized controls $u_i(t)$, such that they guarantee the convergence of the agents to a consensus/formation, i.e.

$$\lim_{t \rightarrow \infty} (x_i(t) - x_j(t)) = 0 \quad \forall i, \forall j$$  

(3)

The approach developed here is motivated by [18] and called Virtual Agent Scheme. In this approach, defining $N$ virtual agents with identical linear dynamics given as

$$\dot{\hat{x}}_i(t) = A\hat{x}_i(t) + Bu_i(t) \quad i = 1, 2, ..., N$$  

(4)

where $\hat{x}_i(t) \in \mathbb{R}^n$ and $\hat{u}_i(t) \in \mathbb{R}^m$ are the state and the control vectors of the $i$th virtual agent, using any developed method for decentralized formation/consensus control of linear agents with identical dynamics, one propagates the state trajectory of each of the virtual agents using the initial conditions $\hat{x}_i(0) = x_i(0)$, $i = 1, 2, ..., N$, and the knowledge of the states of its neighbors. The result is $N$ trajectories $\hat{x}_i(t)$ which converge to the formation/consensus. Having these trajectories, hereafter called reference trajectories, one can design $N$ nonlinear tracking controllers for the actual agents given by (2) to force each one of them to follow the respective reference trajectory, i.e., forcing $x_i(t)$ to track $\hat{x}_i(t)$.

Note that, this scheme results in a decentralized control in the sense that in implementation the $i$th agent, $i = 1, 2, ..., N$, only needs to access the states of its neighbors to propagate its reference trajectory $\hat{x}_i(t)$. Once the reference signal is obtained, the agent follows it using a nonlinear tracking controller.

III. TRACKING SNAC

A. Theory

Consider the nonlinear discrete-time input-affine system

$$x_{k+1} = f(x_k) + g(x_k)u_k \quad k = 0, 1, 2, ...$$  

(5)

where $x_k \in \mathbb{R}^n$ and $u_k \in \mathbb{R}^m$ denote the state and the control vectors at time $k$, respectively. $f(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^n$ and $g(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times m}$ are the system dynamics and the initial states is given by $x_0$.

Given the reference signal $y_k \in \mathbb{R}^n$ with the dynamics of

$$r_{k+1} = F(r_k) \quad k = 0, 1, 2, ...$$  

(6)

$$y_k = C r_k$$  

(7)

where $r_k \in \mathbb{R}^p$ is the state vector of the reference signal propagated by the dynamics $F(\cdot) : \mathbb{R}^p \rightarrow \mathbb{R}^p$ with the given initial value of $r_0$, and $C \in \mathbb{R}^{n \times p}$ is a real valued matrix, the objective is selecting a control history $u_k$, $k = 0, 1, 2, ...$, such that the below cost function is minimized.

$$J = \frac{1}{2} \sum_{k=0}^{\infty} ((x_k - y_k)^T Q (x_k - y_k) + u_k^T R u_k)$$  

(8)

where $Q \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{m \times m}$ are weighting matrices for the states’ errors, and the control effort, respectively. $Q$ is assumed to be a positive definite or positive semi-definite matrix and $R$ is a positive definite matrix.

Augmenting the system state vector with the vector of the reference signal, i.e. $r_k$, leads to defining the augmented state vector of $\zeta_k \in \mathbb{R}^{n+p}$ where

$$\zeta_k \equiv \begin{bmatrix} x_k \\ r_k \end{bmatrix}$$  

(9)

The cost function (8) now becomes

$$J = \frac{1}{2} \sum_{k=0}^{\infty} (\zeta_k^T Q \zeta_k + u_k^T R u_k)$$  

(10)

where

$$Q = \begin{bmatrix} Q & -Q C \\ -C^T Q & C^T Q C \end{bmatrix}$$  

(11)

and the dynamics of the augmented state vector is given by

$$\zeta_{k+1} = \tilde{f}(\zeta_k) + \tilde{g}(\zeta_k)u_k \quad k = 0, 1, 2, ...$$  

(12)
\[ \hat{f}(\zeta_k) \equiv \begin{bmatrix} f(x_k) \\ F(r_k) \end{bmatrix}, \quad \hat{g}(\zeta_k) \equiv \begin{bmatrix} g(x_k) \\ 0 \end{bmatrix} \]  

(13)

Note that the tracking problem has now been converted to a standard regulation problem whose solution is given by the HJB equation below

\[ J^*(\zeta_k) = \min_{u_k} \left( \frac{1}{2} (\zeta_k^T \hat{Q} \zeta_k + u_k^T R u_k) + J(\zeta_{k+1}) \right) \]

\[ k = 0, 1, 2, \ldots \]  

(14)

\[ u_k^* = \text{argmin}_{u_k} \left( \frac{1}{2} (\zeta_k^T \hat{Q} \zeta_k + u_k^T R u_k) + J(\zeta_{k+1}) \right) \]  

(15)

Define the costate vector as \( \lambda_k \equiv \frac{\partial j_k}{\partial x_k} \) where \( f(\zeta_k) \) is denoted by \( J_k \) to get

\[ u_k^* = -R^{-1} \hat{g}(\zeta_k)^T \frac{\partial j_{k+1}}{\partial \zeta_k} = -R^{-1} \hat{g}(\zeta_k)^T \lambda_{k+1} \]  

(16)

Replacing \( u_k \) in (14) by \( u_k^* \), the HJB equation reads

\[ J^*(\zeta_k) = \frac{1}{2} (\zeta_k^T \hat{Q} \zeta_k + u_k^T R u_k^*) + J^*(\zeta_{k+1}) \]  

(17)

The costate equation can be derived by taking the derivative of both sides of (17) with respect to \( \zeta_k \) as

\[ \lambda_{k+1} = \hat{Q} \zeta_{k+1} + \hat{A}_{k+1} \lambda_{k+2} \]  

(18)

where

\[ \hat{A}_{k+1} \equiv \frac{\partial \lambda_{k+2}}{\partial \zeta_{k+1}} = \frac{\partial (f(\zeta_{k+1}) + g(\zeta_{k+1} u_{k+1}))}{\partial \zeta_{k+1}} \]  

(19)

Partitioning \( \lambda_k \) as two separate parts of \( \alpha_k \in \mathbb{R}^n \) and \( \beta_k \in \mathbb{R}^p \) as

\[ \lambda_k \equiv \begin{bmatrix} \alpha_k \\ \beta_k \end{bmatrix} \]  

(20)

and substituting it in (18) along with using (9), (11), and (13) leads to

\[ \begin{bmatrix} \alpha_{k+1} \\ \beta_{k+1} \end{bmatrix} = \begin{bmatrix} Q & -QC \\ -C^T Q & C^T Q C \end{bmatrix} \begin{bmatrix} \alpha_{k+1} \\ \beta_{k+1} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{\partial f(\zeta_{k+1}) + g(\zeta_{k+1} u_{k+1})}{\partial \zeta_{k+1}} \end{bmatrix} \begin{bmatrix} \alpha_{k+2} \\ \beta_{k+2} \end{bmatrix} \]  

(21)

which is actually two decoupled equations

\[ \alpha_{k+1} = Q(x_{k+1} - y_{k+1}) + A_{k+1}^T \alpha_{k+2} \]  

(22)

\[ \beta_{k+1} = -CQ(x_{k+1} - y_{k+1}) + \left( \frac{\partial (r_{k+1})}{\partial r_{k+1}} \right)^T \beta_{k+2} \]  

(23)

and the optimal control equation (16) changes to

\[ u_k = -R^{-1} g(x_k)^T \alpha_{k+1} \]  

(25)

Interestingly, for the optimal control calculation using (25), one does not need \( \beta_{k+1} \) and only \( \alpha_{k+1} \) is required.

Using the SNAC scheme [29], a neural network (NN) is selected to generate \( \alpha_{k+1} \). From (22), the costate vector \( \alpha \) is observed not only to be a function of the states \( x \), but also a function of the reference signal \( y \), hence, the inputs to the network are selected to be \( x_k \) and \( y_k \). Denoting the NN mapping with \( NN(\cdot) \) one has

\[ \alpha_{k+1} = NN(x_k, y_k) \]  

(26)

Selecting the network structure of linear in the tunable weights, one has

\[ \alpha_{k+1} = W^T \phi(x_k, y_k) \]  

(27)

where \( W \in \mathbb{R}^{l \times n} \) denotes the network weight matrix, \( \phi(\cdot) : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R} \) is composed of \( l \) linearly independent scalar basis functions.

The network training target, denoted by \( \alpha^T \), can be calculated using the following equation.

\[ \alpha_{k+1}^T = Q(x_{k+1} - y_{k+1}) + A_{k+1}^T \alpha_{k+2} \quad k = 0, 1, 2, \ldots \]  

(28)

which in the training process, \( \alpha_{k+2} \) on the right hand side of (28) will be substituted by \( W^T \phi(x_{k+1}, y_{k+1}) \). Noting that the close loop dynamics, using (5), (25), and (27), is given by

\[ x_{k+1} = f(x_k) - g(x_k) R^{-1} g(x_k)^T W^T \phi(x_k, y_k) \]  

(29)

equation (28) changes to

\[ \alpha_{k+1} = Q(f(x_k) - g(x_k) R^{-1} g(x_k)^T W^T \phi(x_k, y_k) - CF(r_k)) + A_{k+1}^T W \phi \left( f(x_k) - g(x_k) R^{-1} g(x_k)^T W^T \phi(x_k, y_k), CF(r_k) \right) \]  

(30)

Note that in (30) which is supposed to generate \( \alpha_{k+1} \) to be used for training the weights, the right hand side of the equation is dependent on \( W \), hence, the calculated target \( \alpha_{k+1}^T \) is itself a function of the weights and the optimal target needs to be obtained through a successive approximation scheme, called reinforcement learning.

Defining the training error as

\[ e_{k+1} \equiv \alpha_{k+1} - \alpha_{k+1}^T = W_k^T \phi(x_k, y_k) - \alpha_{k+1}^T \]  

(31)

the iterative process of learning \( W \) can be summarized as the algorithm given below.
Algorithm 1:
1- Randomly select the state vector \(x_k\) and the state vector of the reference signal, i.e. \(r_k\), belonging to the domain of interest.
2- Through the process showed in Fig. 1, calculate \(\alpha^T_{k+1}\).
3- Train network weight \(W\) using input-target pair \((x_k, y_k), \alpha^T_{k+1}\).
4- Calculate the training error \(e_{k+1}\) using (31).
5- Repeat step 1 to 4 until the error \(e_{k+1}\) converges to zero or a small value for different random \(x_k\)'s and \(r_k\)'s selected in step 1.

### Diagram Illustration

![Tracking SNAC training diagram](image)

Having the input-target pair \((x_k, y_k), \alpha^T_{k+1}\) calculated, the network can be trained using any training method [30]. The selected training law in this study is the least squares method. Assume that in each iteration of Algorithm 1, instead of one random state and reference signal, \(q\) random states and reference signals denoted by \(x^{(i)}\) and \(y^{(i)}\), \(i = 1, 2, ..., q\), are selected. Denoting the training target \(\alpha^T\) calculated using \(x^{(i)}\) and \(y^{(i)}\) by \(\alpha^T(x^{(i)}, y^{(i)})\), the objective is finding \(W\) such that it solves

\[
\begin{align*}
W^T \phi(x^{(1)}, y^{(1)}) &= \alpha^T(x^{(1)}, y^{(1)}) \\
W^T \phi(x^{(2)}, y^{(2)}) &= \alpha^T(x^{(2)}, y^{(2)}) \\
& \vdots \\
W^T \phi(x^{(q)}, y^{(q)}) &= \alpha^T(x^{(q)}, y^{(q)})
\end{align*}
\]  

(32)

Define

\[
\alpha^T \equiv [\alpha^T(x^{(1)}, y^{(1)}) \alpha^T(x^{(2)}, y^{(2)}) ... \alpha^T(x^{(q)}, y^{(q)})]
\]  

(34)

Using least squares, the solution to system of linear equations (32) is given by

\[
W = (\phi \phi^T)^{-1} \phi \alpha^T
\]  

(35)

Note that for the inverse of matrix \((\phi \phi^T)\) to exist, one needs the basis functions \(\phi\) to be linearly independent and the number of random states \(q\) to be greater than or equal to the number of neurons, \(l\).

Though (35) looks like a one-shot solution for the ideal NN weights, the training is an iterative process which needs selecting different random states and reference signals and updating the weights through solving (35) successively. Note that \(\alpha^T\) used in the weight update (35), as explained earlier, is not the true optimal costate and is a function of current estimation of the ideal unknown weight, i.e. \(\alpha^T = \alpha^T(W)\). Denoting the weights at the \(i\)th epoch of the weight update by \(W^i\) the iterative procedure is given by

\[
W^{i+1} = (\phi \phi^T)^{-1} \phi \alpha^T(W^i)^T
\]  

(36)

One has

\[
\lim_{i \to \infty} W^i = W^*
\]  

(37)

where \(W^*\) denotes the optimal NN weights in the sense that it generates the optimal costate vector \(\alpha_{k+1}\). For this purpose, one starts with an initial weight \(W^0\) and iterates through (35) until the weights converge. The initial weight can be set to zero or can be selected based on the linearized solutions of the given nonlinear system.

**Comment 1:** If the reference signal is such that there is no control needed after the system achieves perfect tracking, then the term \(u^T_k R u_k\) in the cost function becomes zero. If not, the cost function becomes unbounded. In this case, an alternative formulation is suggested as given below:

Use discounted cost function

\[
J = \sum_{k=0}^{\infty} \gamma^k ((x_k - y_k)^T Q (x_k - y_k) + u^T_k R u_k)
\]  

(38)

with \(0 < \gamma < 1\) being the discount factor. In this case, the costate equation (22) changes to

\[
\alpha_k = Q(x_k - y_k) + \gamma A^T_{k+1} \alpha_{k+1}
\]  

(39)

and the optimal control equation (25) reads

\[
u_k = -\gamma^{-1} R^{-1} g(x_k)^T \alpha_{k+1}
\]  

(40)

### IV. Formation Control Using Tracking SNAC

Following the Virtual Agent Scheme of formation control of nonlinear agents, in this section a consensus protocol developed in the literature for linear identical agents will be used for formation/consensus control of virtual agents. The Tracking SNAC developed in section III will then be used to

...
force each nonlinear agent to follow the respective reference trajectory generated by its virtual agent.

Assume $N$ identical virtual agents with general linear dynamics given below

$$\dot{x}_i = A\dot{x}_i + B\dot{u}_i \quad i = 1, 2, \ldots, N$$ \hspace{1cm} (41)

where $\dot{x}_i \in \mathbb{R}^n$, $\dot{u}_i \in \mathbb{R}^m$ are the state and control vectors of $i$th agent, and $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$ are the selected dynamics for the agents as some design parameters, $n$ and $m$ denote the dimensions of the state space and the input, respectively.

These agents are assumed to communicate with their neighbors only, hence, denoting the set of neighbors of $i$th agent with $N_i$, this agent has access to the states of agent $j$ if and only if $j \in N_i$. The objective is to design a decentralized controller to reach consensus, i.e. to have $(\dot{x}_i - \dot{x}_j) \to 0$ for every $i$ and $j$ between 1 and $N$.

In order to accomplish that, a gain matrix $K \in \mathbb{R}^{m \times n}$ is selected such that the decentralized controller given by

$$\dot{u}_i = \sum_{j \in N_i} K(\dot{x}_i - \dot{x}_j) \quad i = 1, 2, \ldots, N$$ \hspace{1cm} (42)

results in a consensus [10]-[13] and [33].

Defining

$$X \equiv [\dot{x}_1^T \ \dot{x}_2^T \ \ldots \ \dot{x}_N^T]^T$$ \hspace{1cm} (43)

$$U \equiv [\dot{u}_1^T \ \dot{u}_2^T \ \ldots \ \dot{u}_N^T]^T$$ \hspace{1cm} (44)

and using Kronecker products notation [14] for compactness, one has

$$\dot{X} = (I_N \otimes A)X + (I_N \otimes B)U$$ \hspace{1cm} (45)

where $I_N$ is an identity matrix of dimension $N \times N$. The feedback control (42) can be expressed in a compact Kronecker products form as [3]

$$U = (L \otimes K)X$$ \hspace{1cm} (46)

where $L$ is the Laplacian matrix representing the topology of the communication graph [6], [3]. The closed loop dynamics is then given by

$$\dot{X} = (I_N \otimes A + L \otimes BK)X$$ \hspace{1cm} (47)

Using the overall state $X$ which is composed of the states of all of the virtual agents, as the state of the reference signal, i.e. $r$ as used in (6), the reference signal $y$ used in (7) for the agent $i$ will be $\dot{x}_i$ which is a part of $X$. Matrix $C$, as used in (7), will be a matrix with dimension $n \times nN$ and for $i$th agent is given by

$$C = [0_{nN(i-1)n} \quad I_n \quad 0_{nN(n-i)n}]$$ \hspace{1cm} (48)

in which $0_{a\times b}$ , with $a$ and $b$ being non-negative integers, denotes a zero matrix of dimension $a \times b$.

Since the nonlinear tracking controller developed in this paper admits discrete-time reference signals one needs to discretize the closed loop dynamics of $X$ given by (47) to find $F(.)$ as used in (6). Selecting small sampling time $\Delta t$, and denoting discretized $X(t) \by \dot{X}_k$ where $t = k\Delta t$, the discrete closed loop dynamics of $X$ can be given by

$$X_{k+1} = X_k + \Delta t(I_N \otimes A + L \otimes BK)X_k$$ \hspace{1cm} (49)

hence,

$$F(X_k) = X_k + \Delta t(I_N \otimes A + L \otimes BK)X_k$$ \hspace{1cm} (50)

Having $F(.)$ and $C$, one may train as many Tracking SNAC as the number of agents based on the dynamics of the respective agent using Algorithm 1. If the agents’ nonlinear dynamics are identical, one trained neural network is enough and can be duplicated to be used for different agents. Afterward, the Tracking SNAC’s may be used in online implementation along with the decentralized linear controller of the virtual agents in the manner that the latter generates the reference signal and the former tracks it.

V. NUMERICAL ANALYSIS

A. Modeling

For demonstration of the new controller, the problem of several satellites seeking consensus on their attitudes has been selected, in the sense that the satellites need to adjust their attitude with respect to each other while each one of them has access to the attitude of its neighbors only. Satellite dynamics can be represented as [31]

$$\frac{d\omega}{dt} = \mathbb{I}^{-1}(\tau - \omega \times \omega)$$ \hspace{1cm} (51)

where $\mathbb{I}$, $\omega$, and $\tau$ are the inertia tensor, angular velocity vector of the body frame with respect to inertial frame and the vector of the torque applied on the satellite, respectively. All vectors are represented in the body frame and the sign $\times$ denotes cross product of two vectors.

The only applied torque is assumed to be the control torque which is created using satellite actuators. Following [32] and its order of transformation, the kinematic equation of the satellite is

$$\frac{d}{dt} \begin{bmatrix} \varphi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} 1 & \sin(\varphi)\tan(\theta) & \cos(\varphi)\tan(\theta) \\ 0 & \cos(\varphi) & -\sin(\varphi) \\ 0 & \sin(\varphi)/\cos(\theta) & \cos(\varphi)/\cos(\theta) \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$ \hspace{1cm} (52)

where $\varphi, \theta$, and $\psi$ are the three Euler angles describing the attitude of the satellite with respect to $x, y, \text{and } z$ axes of the inertial coordinate system, respectively. The subscript $x, y, \text{and } z$ denote the corresponding elements of the vector $\omega$.

To form the state space equation of the satellite attitude control problem, one can choose the three Euler angles and the
three elements of the angular velocity as states and describe the following state space equation

\[ \dot{x} = f(x) + g(x)u \quad (53) \]

where

\[ f(x) \equiv \begin{bmatrix} M_{3 \times 1} \end{bmatrix} \begin{bmatrix} I^{-1}(-\omega \times I\omega) \end{bmatrix} \quad (54) \]

\[ g \equiv \begin{bmatrix} 0_{3 \times 3} \end{bmatrix} \begin{bmatrix} r^{-1} \end{bmatrix} \quad (55) \]

\[ x = [\varphi \theta \psi \omega_x \omega_y \omega_z]^T \quad (56) \]

\[ u = [\tau_x \tau_y \tau_z]^T \quad (57) \]

\( M_{3 \times 1} \) denotes the right hand side of equation (52) and \( 0_{3 \times 3} \) denotes a three-by-three null matrix.

Assuming a network of four satellites, with identical dynamics for simplicity, and respective state vector and control vectors of \( x_i \) and \( u_i \) for \( i = 1, 2, 3, 4 \), one has

\[ \dot{x}_i = f(x_i) + g(x_i)u_i \quad i = 1, 2, 3, 4 \quad (58) \]

The moment of inertia matrix of the satellites is chosen as

\[ \mathbb{I} = \begin{bmatrix} 110 & 2 & 5 \\ 2 & 100 & 1 \\ 5 & 1 & 90 \end{bmatrix} \text{kg.m}^2 \quad (59) \]

and the communication topology showed in Fig. 2 and represented by \( L \) is given by

\[ L = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \quad (60) \]

![Fig. 2. The graph representing the communication between the satellites.](image)

The initial Euler angles \( [\varphi, \theta, \psi] \), in the unit of degrees, are selected as \([-40, 60, 0], [40, 30, 20], [30, 40, 10], \) and \([10, -20, -70]\), and the initial angular rates \( \omega_x, \omega_y, \omega_z \) in radian per second are selected as \([0.01, 0.01, 0.01], [0.01, -0.01, -0.01], [0.01, -0.01, 0.01], \) and \([-0.01, 0.01, 0.01]\), for satellite 1 to 4, respectively.

Simulation of the formation of virtual agents is shown in Fig. 3 using the dashed plots. As can be seen, the attitudes of the four satellites have reached a consensus, i.e. through some slew maneuvers, all of them are pointing in the same direction as desired at the end of the simulation.

The Tracking SNAC design and training is the next step. The sampling time of \( \Delta t = 0.1 \text{ sec.} \) is selected for discretizing the continuous dynamics (58) and denoting the basis functions inputs with \( x \) and \( y \) and their elements with \( x_i \) and \( y_i \), \( i = 1, 2, ..., 6 \), the vector \( \Phi(x, y) \) is selected of elements \( x_i^a \) and \( y_i^a \) for \( a = 1, 2, i = 1, 2, ..., 6 \) and \( j = 1, 2, 3 \), along with terms \( x_i y_j \) for \( i = 1, 2, 3 \) and \( j = 1, 2, 3 \). This leads to \( \Phi(x, y) \in \mathbb{R}^{6^6} \).

The first three elements of \( x \) and \( y \) are the Euler angles with the unit of radians, while the last three are the rates with the unit of radian per second. For faster training, it is advisable to normalize the rates by multiplying them with a factor, selected as 100 in this study, to end up with network inputs all being almost of the same order of magnitude.

The state and control weight matrices for the cost function (8) are selected as

\[ Q = \text{diag}(100 \ 100 \ 100 \ 0 \ 0 \ 0) \quad (64) \]

Since the above \( A \) matrix is unstable, one may add three negative values to the last three diagonal elements of the matrix to make the rates stable, resulting in a neutrally stable system, i.e. the rates converge to zero while the Euler angle converge to some constant values. Note that the objective of the given formation problem is having the satellites to align their attitudes, regardless of what the final attitude, upon which they have reach consensus is. One may select an \( A \) matrix for the virtual agents as

\[ A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad B = g \quad (62) \]

\[ A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.5 \end{bmatrix} \quad (63) \]

For the purpose of designing the consensus protocol (42) the method developed in [33] is used in this simulation. Note that, any other consensus protocol like those developed in [10]-[13] could also be used for this step to generate the reference signals.

The Tracking SNAC design and training is the next step. The sampling time of \( \Delta t = 0.1 \text{ sec.} \) is selected for discretizing the continuous dynamics (58) and denoting the basis functions inputs with \( x \) and \( y \) and their elements with \( x_i \) and \( y_i \), \( i = 1, 2, ..., 6 \), the vector \( \Phi(x, y) \) is selected of elements \( x_i^a \) and \( y_i^a \) for \( a = 1, 2, i = 1, 2, ..., 6 \) and \( j = 1, 2, 3 \), along with terms \( x_i y_j \) for \( i = 1, 2, 3 \) and \( j = 1, 2, 3 \). This leads to \( \Phi(x, y) \in \mathbb{R}^{6^6} \).

The first three elements of \( x \) and \( y \) are the Euler angles with the unit of radians, while the last three are the rates with the unit of radian per second. For faster training, it is advisable to normalize the rates by multiplying them with a factor, selected as 100 in this study, to end up with network inputs all being almost of the same order of magnitude.

The state and control weight matrices for the cost function (8) are selected as

\[ Q = \text{diag}(100 \ 100 \ 100 \ 0 \ 0 \ 0) \quad (64) \]
The objective is to force the Euler angles to follow those of the virtual agents, hence, the first three diagonal elements of matrix $Q$, which are the penalty on Euler angles’ tracking errors, are selected high, while the last three are set to zero to let the tracking controller be free in selecting suitable rates to result in the best Euler angles tracking.

Note that the cost function (8) is well-posed and finite, since no control is needed after the attitudes of all the satellites reach a steady state value. (See Comment 1).

The training is done for 300 epochs, where in each iteration, 100 random states and reference signals are selected to perform the least squares given in (35). The weights of the Tracking SNAC converged after around 150 iterations of Algorithm 1 as seen in Fig. 4 which shows the history of evolution of the weights versus the training iterations.

Once trained, the neurocontroller was used for controlling the satellites with actual nonlinear dynamics. Since the satellites are selected to have identical dynamics, one trained network can be duplicated to four networks for controlling four separate satellites. In Fig. 3, the histories of the Euler angles for the actual satellites are plotted using solid plots. As seen, the Tracking SNAC’s have been able to force the Euler angles of each individual satellite to track the respective reference signal. The net result is the convergence of the attitude of all of the four satellites to the same direction using a decentralized controller.

Comparing the angles histories of the virtual and actual agents, it can be seen that in the beginning of the maneuvers, the tracking errors tend to grow and then they gradually decrease. The initial growth is because of the fast changes in the Euler angles, which is easy to be done for the virtual agents without inertia, while for the actual satellites it takes a while to change the angles and go toward the consensus attitude.

VI. CONCLUSIONS

Decentralized formation control of agents with heterogeneous nonlinear dynamics is done using virtual agent scheme through generating some reference trajectories to be followed by the nonlinear agents. The reference trajectory generated using a decentralized controller guarantees the convergence of the virtual agents to the consensus/formation, hence, tracking it leads to convergence of the actual agents to the desired formation. The problem of tracking of nonlinear systems was then solved by developing a neural network which optimally tracks a reference signal. The simulation study showed the promising performance of the developed formation controller.

REFERENCES


